Thesis

Intuitive and Controllable AI Steering Interfaces

David Chuan-En Lin

Human-Computer Interaction Institute

School of Computer Science

Carnegie Mellon University

Pittsburgh, PA 15213

Thesis Committee

Dr. Nikolas Martelaro, Chair, CMU

Dr. Aniket Kittur, CMU

Dr. David Lindlbauer, CMU

Dr. Michael Terry, Google DeepMind

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Keywords: Interaction Techniques, Machine Learning, Design, Human-AI Interaction, Creativity Support Tools, Authoring Tools, Interactive AI

Abstract

A grand challenge in computer science is transforming AI from passive automation machines into interactive systems that users can actively steer and control. AI steering interfaces are a key pillar in realizing this goal, giving users a controllable interfacing layer to guide AI behavior. Current AI interaction methods - such as text prompts and basic parameter sliders - offer initial glimpses of this potential, providing rudimentary ways to influence AI systems. While these elementary approaches already deliver value to millions of users worldwide, substantial opportunities remain for more expressive control mechanisms. Yet designing effective steering interfaces faces a fundamental trade-off between intuitiveness and controllability. As interfaces become more intuitive (one-click solutions, text boxes), control diminishes, while interfaces that maximize control (programming, complex editors) become expert-only tools. Most conventional approaches fall along this spectrum, limiting users to either ease of use or expressive control.

My research aims to design AI steering interfaces that break the traditional intuitiveness-controllability trade-off. By systematically mapping how people interact (HCI interface paradigms) to how models can be controlled (controllable mechanisms of AI models), I identify promising combinations that shift toward high intuitiveness and high controllability simultaneously. I introduce six systems demonstrating this mapping: Soundify pairs object-centric control with class activation maps, VideoMap combines map-based overviews with latent space navigation, Jigsaw connects flow-based interfaces to model orchestration, PseudoClient links example-based interaction to few-shot meta-learning, Inkspire blends sketch-based iteration with structural conditioning, and StyleGenome (proposed work) integrates evolutionary interfaces with parameter-efficient adaptation. These systems demonstrate that expressive AI control can be made both intuitive for novices and powerful for experts, enabling graphic designers to personalize AI using their own work, video editors to navigate large-scale video content spatially, and product designers to iterate through broad design spaces by sketching.

Contents

I. Introduction

1.1 Defining AI Steering Interfaces

The desire to steer computational systems dates back to the earliest interactive computers. Sutherland's seminal Sketchpad [Sutherland 1963] introduced direct manipulation - users could interact with visible objects and receive immediate feedback, making complex behaviors controllable through interaction rather than programming. As artificial intelligence developed, early systems adapted this philosophy for intelligent applications. For example, Winograd's SHRDLU [Winograd 1968] allowed users to control a virtual blocks-world agent through natural conversation. Today's steering has evolved beyond dialogue and rule editing to manipulating learned representations directly: users now guide AI models by crafting prompts, adjusting parameter sliders, and navigating feature spaces.

"AI steering interfaces" are user-facing tools that provide explicit control mechanisms for directing AI model behaviors through interaction. An effective steering interface combines intuitiveness (being easy to use and interact with) with controllability (allowing users to meaningfully guide and shape the AI's behavior). By transforming opaque AI models into controllable capabilities, steering interfaces enable human-AI collaboration across a wide range of creativity and productivity applications.

1.2 The Intuitiveness-Controllability Spectrum

AI steering interfaces can be arranged along a spectrum of intuitiveness - how easily people, especially non-experts, can use and interact with them. Figure 1 shows exemplary interface paradigms along this spectrum. At one end sit expert-oriented mechanisms like programming and timeline editors that demand specialized knowledge or training. Moving toward higher intuitiveness, sliders and dials lower barriers while still exposing meaningful parameters. Text prompts further reduce entry costs by letting users state goals in natural language. At the far end, one-click solutions conceal configuration entirely.

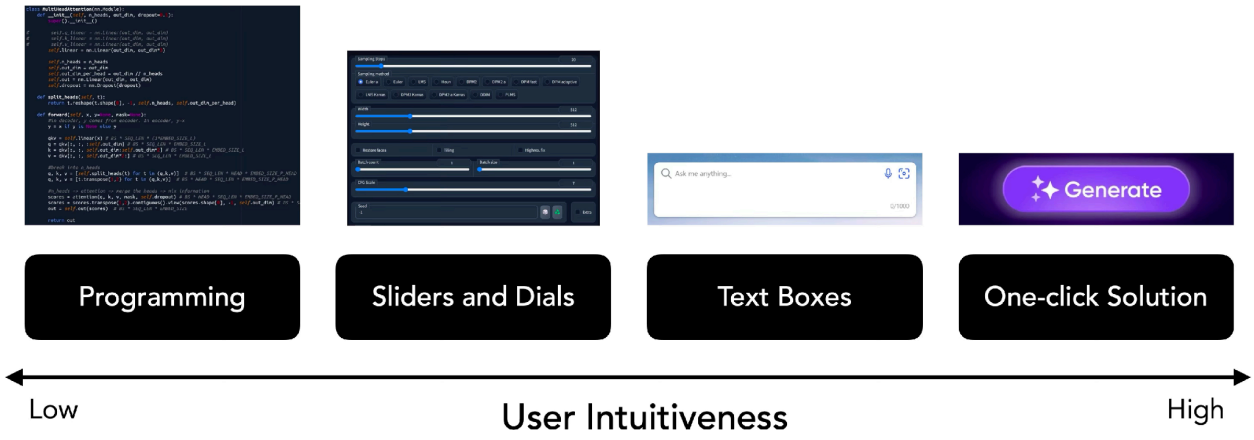

Figure 1: Intuitiveness spectrum of AI steering interfaces (from low to high: programming, sliders and dials, text boxes, and one-click solution).

However, there is a second critical dimension: controllability. Controllability is multifaceted - a controllable system can exhibit some of the following characteristics:

- Personalized: Ability to adapt to individual preferences (generic → personalized)
- Responsive: Speed of feedback loops (slow → real-time)
- Expressive: Range of outcomes the AI can produce (narrow → broad)
- Contextual: Ability to consider and respond to context (context-blind → context-aware)
- Inspectable: Ability to easily explore the outcome space (constrained view → overview)
- Specific: Level of granularity users can control (coarse → fine-grained)
- Extensible: Ability to combine elements modularly (monolithic operation → modular building blocks)

In this thesis, I refer to these as the PRECISE characteristics of AI steering controllability.

Ideally, we want interfaces that are broadly intuitive and precisely controllable. In practice, they often trade off. As interfaces become more intuitive (one-click, prompt-only), control can diminish. As interfaces maximize control (programming, complex timeline editors), they become expert-only tools. Plotting intuitiveness versus controllability as a design space (Figure 2) reveals that conventional approaches cluster along an inverse diagonal - increases in one dimension come at the expense of the other.

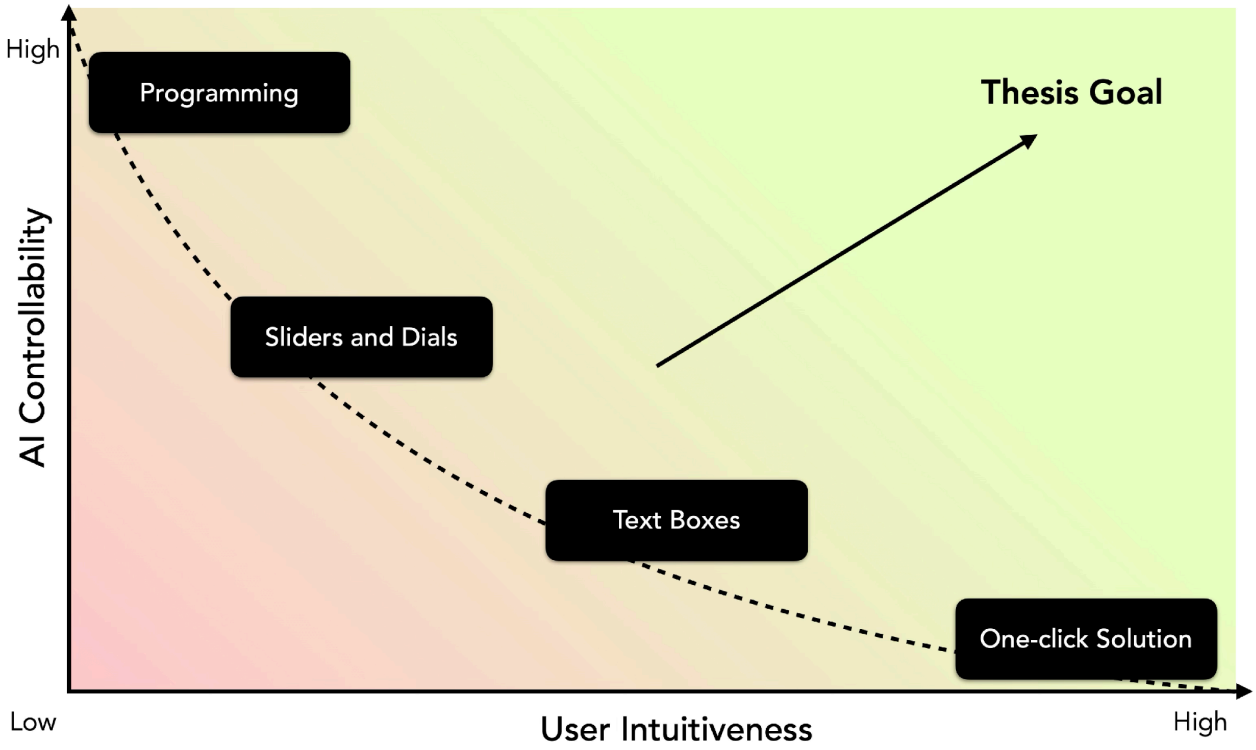

Figure 2: A design space plotting intuitiveness vs. controllability. Most approaches lie along a diagonal where one increases as the other decreases. Ideally, we want interfaces that offer both high intuitiveness and high controllability (green region).

This inverse relationship between intuitiveness and controllability reflects a tension in interface design. Intuitive interfaces succeed by establishing strong conventions and constraints that guide users toward successful outcomes. Controllable interfaces succeed by exposing flexibility and options that allow users to achieve precise results. Traditional design wisdom suggests these goals are fundamentally at odds: constraints that help novices limit experts, while flexibility that empowers experts overwhelms novices.

However, this thesis challenges this traditional wisdom by recognizing that the apparent trade-off stems from a mismatch between interface paradigms and underlying control mechanisms. When we force users to control AI through generic interfaces like text boxes or sliders, we're asking them to translate their intentions through inappropriate channels. This thesis argues that AI steering interfaces can shift towards the region of high intuitiveness and

high controllability by systematically mapping how people interact (interface paradigms) to how models can be controlled (model control mechanisms). The goal of this thesis is to advance this shift.

1.3 Thesis Organization

In this thesis, I focus on designing intuitive and controllable AI steering interfaces. Chapter 2 introduces my core research methodology, which involves mapping the design space between HCI interface paradigms and controllable mechanisms in AI models. This chapter provides background on the interface paradigms and AI control mechanisms most relevant to my work.

Applying my research methodology, I have designed and built a series of novel AI steering interfaces. Chapter 3 presents work that focuses on raising intuitiveness while preserving or improving controllability, including Soundify (Chapter 3.1), VideoMap (Chapter 3.2), and Jigsaw (Chapter 3.3). Chapter 4 introduces work that focuses on raiseing controllability while preserving or improving intuitiveness, including PseudoClient (Chapter 4.1) and Inkspire (Chapter 4.2). For each system, I describe how it improves specific PRECISE characteristics of AI steering controllability and where it falls on the intuitiveness-controllability spectrum, demonstrating shifts toward the targeted region of high intuitiveness and high controllability.

Chapter 5 proposes StyleGenome, which explores pairing a DNA genetic evolution interface metaphor with underlying AI model control mechanisms built on low-rank adaptation techniques. This chapter includes an introduction, followed by system design, evaluation plan, and timeline.

II. Mapping the Design Space: Interface Paradigms × Controllable Mechanisms of AI

This thesis proposes that designing better AI steering interfaces - ones that achieve both high intuitiveness and high controllability - requires systematically mapping the design space between HCI interface paradigms and controllable mechanisms in AI models. This mapping process reveals underexplored combinations that can break the traditional intuitiveness-controllability trade-off. Controllable AI mechanisms provide the technical foundation for precise steering, while intuitive HCI interface paradigms provide intuitive interaction models.

Here I review literature most directly related literature to this thesis's efforts from both domains: interface paradigms that make systems steerable for users, and algorithmic mechanisms that expose control surfaces on AI models.

2.1 HCI Interface Paradigms for Steering

Direct manipulation interfaces

Direct manipulation emphasizes visible objects and predictable outcomes; it remains a key paradigm for making intelligent systems controllable. Grounded in Shneiderman's call for comprehensible, predictable, controllable interfaces, using object-level representations as tangible handles [Shneiderman 1997], the principles focus particularly on object-centric manipulation where users can interact directly with individual elements rather than abstract parameters. The key design principle is making computational objects tangible and directly manipulable to create intuitive control surfaces.

Map-based interfaces

Maps afford overview, zoom, and filter operations over large-scale data. Shneiderman's Visual Information-Seeking Mantra [Shneiderman 1996] formalized the interaction loop that many

interactive visual analytics systems adopt. Zoomable user interfaces such as Pad++ operationalized multiscale navigation and semantic zooming for wayfinding [Bederson, 1994]. The key design principle is providing spatial overview and navigation affordances that enable users to understand relationships and find their way through complex information spaces.

Flow-based interfaces

Dataflow and node-based authoring make pipelines tangible and composable through modular building blocks. LabVIEW popularized graphical dataflow to orchestrate sensing/analysis; the paradigm remains influential in education and instrument control. [LabVIEW]. Web mashup tools such as Yahoo Pipes, Popfly democratized end-user pipeline composition. In IoT, Node-RED operationalizes visual flows for APIs and events, illustrating how nodes and wires lower the floor for non-experts. [Node-RED]. The key design principle is modularity - breaking complex workflows into discrete, reusable components that can be visually connected and reconfigured.

Example-based interfaces

Example-based interfaces allow users to specify desired behavior through concrete instances rather than abstract descriptions. Programming-by-demonstration (PBD) systems such as Watch What I Do [Watch What I Do] and CoScripter [CoScripter] represent one approach to example-based interfaces, but the broader category encompasses any interface where users provide examples to communicate intent. Interactive ML tools like Google's What-If Tool [Google What-If] brought visual debugging to everyday ML practice. The key design principle is enabling users to communicate intent through concrete examples rather than abstract specifications.

Sketch-based interfaces

Sketching supports ambiguity, rapid iteration, and early feedback. SILK [SILK] and DENIM [DENIM] established sketching as a first-class interface for prototyping UIs and websites. Output-directed programming like Sketch-n-Sketch links drawings to code, pointing to bidirectional, live coupling between intent and artifact. The key design principle is supporting

ambiguous, iterative expression that enables rapid exploration and refinement of ideas through natural drawing gestures.

Evolution-based interfaces

Evolutionary-based interfaces allow users to guide an evolutionary algorithm by selecting which "generations" or variants they prefer, rather than specifying exact parameters or goals. The seminal graphics work by Karl Sims demonstrated artificial evolution for visual forms and creatures [Sims, 1994]. Picbreeder scaled this to collaborative online evolution [PicBreeder]. The key design principle is combining random variation to generate unexpected possibilities with human curation and guidance to steer the search toward desirable outcomes.

2.2 Controllable Mechanisms in AI Models

Activation maps

Class Activation Mapping [CAM] and Grad-CAM [Grad-CAM] localize regions corresponding to specific object classes by revealing where models focus attention during inference. Attention visualization methods [Attention maps] expose how transformers [Transformers] weight different input elements during processing. The key capability is steering through detecting spatial or temporal regions in pixels, tokens, or spectrograms that correspond to specific objects or classes.

Latent space navigation

GAN [GAN] and diffusion model latents contain semantically smooth directions that can be exploited for editing. StyleGAN [StyleGAN] first established disentangled style controls while works such as GANSpace [GANSpace], InterFaceGAN [InterFaceGAN], and SeFa [SeFa] discovered interpretable axes for pose, expression, and attributes. StyleCLIP [StyleCLIP] aligns latent edits with CLIP semantics for text-guided manipulation. The key capability is steering through manipulation of learned representations along directions that correspond to human-interpretable concepts.

Orchestrating models

Composition and coordination of foundation models allows creating complex workflows that leverage the complementary strengths of different pre-trained models. Tools such as LangChain [LangChain], Promptchainer [Promptchainer], and ChainForge [ChainForge] provide the ability to chain language model pipelines with constraints and control flow. Frameworks like Toolformer [Toolformer] demonstrate how models can learn to coordinate with external specialized models. The key capability is steering through modular composition and orchestration of foundation model capabilities.

Meta-learning

Meta-learning [Meta-learning] enables models to rapidly adapt to new tasks with minimal data by learning distance metrics between examples. Prototypical Networks learn metric spaces for classification via prototype distances [Prototypical Networks]. Contrastive learning [Contrastive learning] approaches like SimCLR maximize similarity between positive pairs while minimizing it for negative pairs [SimCLR]. The key capability is steering by adapting of AI models using only a few new examples.

Structural conditioning

Diffusion models [Diffusion model] support spatial control using conditioning adapters. ControlNet [ControlNet] augments text-to-image models with auxiliary condition branches such as edges, depth, pose, and sketches. GLIGEN [GLIGEN] allows conditioning via object layouts. The key capability is steering through additional spatial and structural constraints, like reference sketches, depth maps, or pose estimates.

Parameter-efficient adaptation

Parameter-efficient finetuning (PEFT) is a finetuning technique that reduces trainable parameters while matching full finetuning performance. Low-Rank Adaptation (LoRA) [LoRA] and variants [DoRA] allows accelerated finetuning of lightweight task-specific models. DreamBooth binds unique identifiers for customizable subjects and styles and IP-Adapter

expands style composition capabilities. The key capability is steering through lightweight model customization and composable style adaptation.

2.3 Mapping Controllable Mechanisms to Interface Paradigms

Prior work has made valuable contributions in both domains, with each community developing deep expertise in their respective areas. The HCI community has excelled at creating human-centered interfaces around AI systems. These efforts have focused primarily on high-level interaction surfaces like text prompts and less on deep integration of native controllable mechanisms of AI models. The ML community has advanced sophisticated control mechanisms within AI models, developing powerful systems like ControlNet that respond to structural guidance and enable precise model steering. These systems typically employ straightforward interface approaches like sliders to expose their capabilities with less support for rich and expressive interactions.

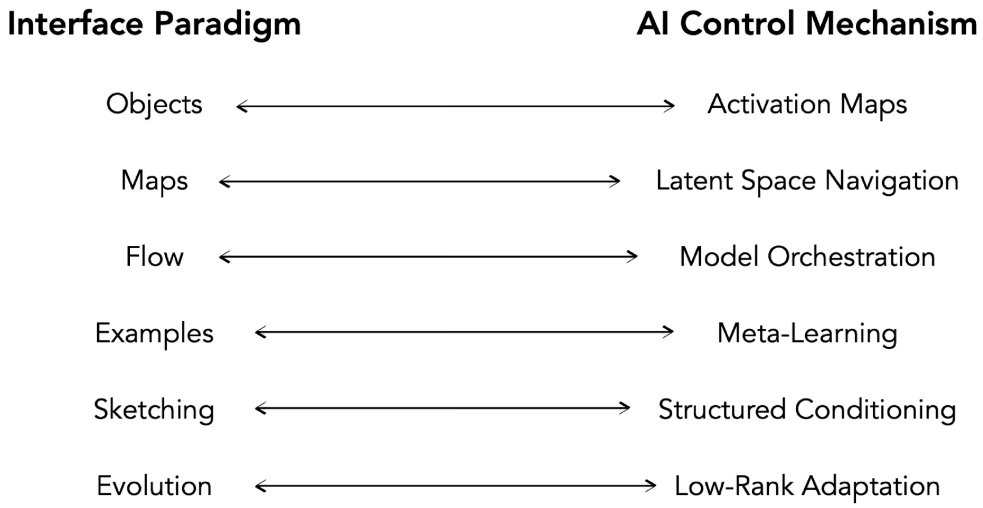

Figure 3: Mapping between HCI interface paradigms and AI control mechanisms explored in this thesis.

This thesis outlines my efforts to bridge this gap by explicitly mapping HCI interface paradigms to controllable AI mechanisms. Each pairing leverages the strengths of both domains: familiar interaction patterns provide intuitiveness, while sophisticated control mechanisms enable

precise steering. In the following sections, I detail five systems I have built over the course of my PhD and a proposed system, that demonstrate specific instantiations of this mapping approach:

- **Soundify** visualizes *activation maps* in AI models to localize sound objects in video, creating an *object-centric* interface that helps video editors easily match sound effects to video by treating different sound sources as individual operable sound objects (Chapter 3.1).
- **VideoMap** projects video frames into *semantically meaningful latent spaces*, creating a *map* interface that helps video editors gain a broad overview of their video assets and easily identify optimal transition points for seamless match cuts (Chapter 3.2).
- **Jigsaw** *orchestrates pre-trained foundation models* using a *puzzle-piece* interface, enabling designers to create complex multimodal AI workflows without programming knowledge by chaining AI model puzzle pieces (Chapter 3.3).
- **PseudoClient** trains an AI model of their personal style with *meta-learning* from a handful of *examples given by a graphic designer*, supporting applications such as personalized design search, feedback, and generation (Chapter 4.1).
- **Inkspire** leverages *structured guidance* and AI-generated sketch scaffolding, allowing product designers to use *iterative sketching* to control AI-generated design, creating a more co-creative process than text prompting (Chapter 4.2).
- **StyleGenome**, my proposed work, aims to build on *low-rank adaptation techniques* to finetune AI image generation models toward finegrained design styles, using the metaphor of *DNA and genetic operations* as control mechanisms to mix, mutate, and evolve creative designs and rapidly explore a broad design space (Chapter 5).

III. Raising Intuitiveness while Preserving Controllability

Creating media content often requires mastering complex professional tools for sound, video, and multimedia editing. In this section, I prioritize intuitiveness - lowering the floor for non-experts - by leveraging AI to enable media editing through familiar interface primitives. Soundify enables object-centric sound design by allowing users to attach sound effects to on-screen objects through visualizing AI model activation maps. Videomap supports video editing on a map, letting users create transitions by identifying neighboring frames in a latent space generated from projected video frame embeddings. Jigsaw facilitates modular multimedia creation by letting users combine AI capabilities like puzzle pieces through multimodal model orchestration. Together, these works demonstrate how exploiting control mechanisms in AI models can transform media content into high-level manipulable primitives that make media creation more approachable for novices.

3.1 Soundify: Sound Objects as an Interface

This chapter was adapted from my published paper: Lin, D. C. E., Germanidis, A., Valenzuela, C., Shi, Y., & Martelaro, N. (2023, October). Soundify: Matching sound effects to video. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (pp. 1-13).

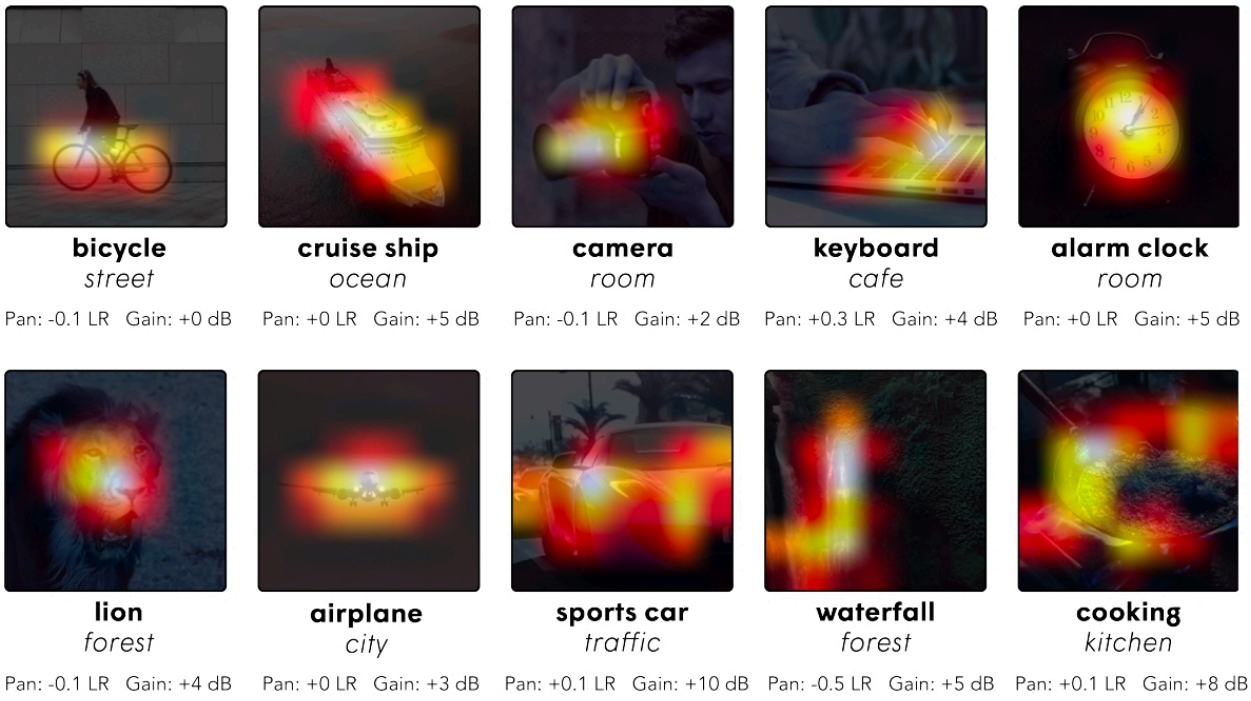

Figure 3.1: Soundify assists users in matching sound effects (in bold) and ambients (in italics) to video, and helps dynamically adjust panning and volume by localizing "sound emitters."

3.1.1 Introduction

“Sound is half the experience in seeing a film.”

— George Lucas, Film Director

In the art of video editing, sound helps add character to an object and immerse the viewer within a space. Although a video's soundscape may be built by sounds recorded on set, it is common for editors to replace or add complementary sounds from external sound collections. The process of adding sounds to video is called the foley pass [1]. During the foley pass, a skilled

video editor analyzes the scene and overlays sounds, such as effects (e.g., bicycle bell ring) and ambients (e.g., street noise). Through formative interviews with 10 professional video editors, we found that this process can be challenging and time-consuming, especially as the amount of video footage scales up. We thus aim to develop a system to assist editors in adding sound to videos.

Many approaches have been proposed to synthesize audio for videos. For example, Visual to Sound [44] learns to generate waveform audio given a visual input and AutoFoley [18] learns to synthesize foley sounds for video clips. However, these works necessitate extensive training of audio generation models over large datasets and often produce subpar sounds containing undesirable noise and artifacts. Our work takes a different approach by leveraging existing studio-quality sound effects libraries. These libraries contain sound clips recorded by experts using first-rate equipment under pristine conditions. Rather than synthesizing audio from scratch, we investigate the alternative approach of retrieving matching high-quality sound clips, then remixing them (i.e., pan and volume parameters) to fit the video footage (Figure 3.6). Our key insight is to extend CLIP [28], an image classification neural network that has learned a joint-embedding space between image and text, into a "zero-shot sound detector" through probing the model's activation maps (Figure 3.1). Given a library of sound clips, we compare the labels of the sound clips to the video to identify the sound(s) that are present in the video. Next, we tune the left-right panning according to the activation map coordinates and the volume according to the activation map area.

In this paper, we present Soundify, a system that assists video editors in matching sound effects to video. From our interviews with professional video editors, we distill four key design principles for Soundify to assist editors—surface, synchronize, spatial, and stack. Given video footage, Soundify helps the video editor surface matching sound clips, synchronize the sound clips to the objects in the video, dynamically adjusts the spatial aspect of the sound clips (panning and volume) based on the video content, and allows the stacking of multiple sound clips such as foreground objects and ambients to create an immersive soundscape. We test the

capability of Soundify against a baseline by matching a wide variety of sounds to complex videos and collecting human ratings from Mechanical Turk. Then, we evaluate the usefulness of Soundify in an expert study with video editors against a baseline, demonstrating improvements in workload, usability, and task completion time. We encourage you to have a look at, or better yet, have a listen to the results of Soundify at https://chuanenlin.com/soundify.

In summary, this paper makes the following contributions:

- Soundify, a system that assists video editors in matching sounds to video.
- A human evaluation (N=889 raters) testing the functionality of Soundify against a baseline. Participants report a significant preference for Soundify's results.
- Insights from an expert study (N=12 professional video editors) with Soundify against a manual editing baseline. Participants experience lighter workload, lower task completion time, and higher usability.

3.1.2 Related Work

Our work is situated among literature in two main branches of research in matching sounds to video: audio synthesis and audio-visual correspondence learning. Further, we refer to prior HCI works on systems to assist users with audio editing.

Audio Synthesis

Audio synthesis is an actively explored topic in the audio research community. Research in this space typically adopts generative models to synthesize raw audio. [25] introduces a deep neural network for synthesizing waveform speech from scratch by training on tens of thousands of audio samples in an autoregressive manner. [14, 23] learn to synthesize coherent waveform audio with Generative Adversarial Networks (GANs). Rather than synthesizing waveform audio, [42] trains a GAN to generate MIDI music notes. [13, 22] explore other types of generative architectures for raw audio generation, including a multi-scale VQ-VAE and a diffusion-based model.

Among works in audio synthesis, several works investigate generating audio specifically based on visual input. [10, 11, 26, 44] explore generating raw audio based on video frames. [15, 36] introduce systems that synthesize plausible music based on videos of people performing musical instruments. [6, 37] learn to generate music driven by human skeleton key points. More recently, [18, 19] learn to synthesize foley-like audio tracks from videos by training on a combination of self-recorded foley clips and YouTube videos. Nonetheless, generative audio synthesis approaches require large amounts of curated data. In addition, generated waveform audio results may contain noise artifacts. On the other hand, professional video editing workflows have high sound quality requirements and typically use clean, studio-recorded sounds. In this work, rather than generating raw audio from scratch, we retrieve audio clips from high-quality audio libraries, then tune the panning and volume of the audio based on the video.

Audio-Visual Correspondence Learning

More recently, researchers have investigated the task of learning audio-visual correspondence (i.e., learning the association between images and audio), typically from large labeled datasets. The audio-visual correspondence learning task was first defined by [7] where the authors train separate visual and audio networks with simple late fusion layers to determine whether an audio-visual pair has correspondence or not. Newer works learn audio-visual correspondence at scale by exploiting the natural synchronization of video frames and audio present in web videos [16, 17, 33, 38, 43]. As audio-visual pairs harvested from videos are noisy in terms of quality, researchers often adopt a weakly-supervised training approach with massive amounts of videos (e.g., several thousands to millions of videos). In this work, we bypass the need to learn audio-visual correspondence from large-scale data by leveraging libraries of labeled audio clips and utilizing the image-text correspondence capability of CLIP [28], a neural network trained on web-scale image-caption pairs. We further extend CLIP to support the fine-grained localization of sound emitters by exploiting activation maps [32], which models where the neural network is "looking at", to achieve spatial audio.

Audio-Related Multimedia Editing in HCI

HCI researchers have developed many tools for audio-related multimedia editing. Soundify builds on a large thread of work that leverages algorithmic and AI techniques to help users complete creative tasks more easily and effectively. [41] helps users add visuals to travel podcasts by mining geographic locations and descriptive keywords. [40] proposes interaction techniques that enable users to add graphical content based on a script or outline. [34] introduces a workflow that combines the editing processes of script writing and audio recording for creating podcasts. [39] helps users automatically shorten voice recordings given a length constraint to cater to different social media platforms. [30] assists users in composing audio stories with a transcript-based speech editing tool and an emotion-driven music browser. [29] helps users generate emotionally relevant music scores for audio stories. [12] enables novice creators to author live audio-driven animations. [35] presents an interface for isolating and selecting specific sounds within audio mixtures. This paper aims to contribute to this thread of work focused on developing systems that augment the capabilities of creators. While many current systems primarily assist users by analyzing textual content, such as scripts, we explore an approach that focuses on analyzing the visuals rather than the text of the content.

Table 1: Interviewee information. We assign an ID to each interviewee (later referenced in Section 4). We list each interviewee's years of professional video and audio editing experience, their preferred suite of editing tools, and a short biography.

ID	Experience	Video and Audio Editing Tool(s)	Short biography
I1	20 years	Adobe Premiere Pro, Avid Pro Tools	Filmmaker, producer, works on feature films, animation, and virtual reality
I2	15 years	Adobe Premiere Pro, Avid Pro Tools	Filmmaker, producer, works on feature films, documentaries, and interactive media
I3	13 years	Adobe Premiere Pro, DaVinci Resolve	Video editor, VFX artist, director, owns a production studio, YouTuber (9M subs)
I4	11 years	Adobe Premiere Pro, Adobe After Effects, Timebolt	Video editor, VFX artist, worked in an advertising agency, YouTuber (300K subs)

I5 12 years Adobe Premiere Pro, Adobe Audition Video editor, VFX artist, worked in software engineering, Instagram creator (400K followers)

I6 15 years Adobe After Effects Video editor, VFX artist, animator, skateboarder, worked with advertising agencies and athletes

I7 20 years Adobe Premiere Pro, Final Cut Pro Video editor, creative director, worked on AAA video game videos and government campaigns

I8 8 years Final Cut Pro, iMovie Video editor, comedian, works on commentaries, former Viner, YouTuber (4M subs)

I9 9 years Adobe Premiere Pro Video editor, animator, programmer, works on tech videos and tutorials, YouTuber (50K subs)

I10 11 years Avid Pro Tools Audio editor, programmer, worked in video game sound design and at an AI research lab

3.1.3 Formative Study

To formulate the design principles for Soundify, we conduct a set of formative interviews with 10 professional video editors to understand their creative processes and uncover potential areas for improvement in their sound editing work. Below, we describe our participants and interview procedure. We then describe the findings from the interviews as part of our discussion on the Design Principles in Section 4.

Participants

We recruited 10 professional video editors (I1 - I10, 2 female, 8 male) from known contacts and social media postings. Our interviewees have varying years of video and audio editing experience (mean=13.80, SD=4.29), work in a variety of roles (e.g., filmmaker, visual effects artist, comedian, programmer, content creator), and work with various types of video (e.g., feature films, commercials, documentaries, animation, online videos) (Table 1). All of our participants actively edit video for their work and add sound effects to enhance the content that they create.

Procedure

We conduct our interviews over video conference. Our interviews consist of three stages. First, we ask the participant to provide an overview of their video and audio editing experience. Second, we ask the participant to describe their typical workflow of adding sounds to video. Finally, we ask the participant to reflect on potential features they would like to see in future video and audio editing tools. We record the interviews and also take notes throughout the interviews. The interview lasts for approximately 30 minutes.

3.1.4 Design Goals

We analyze our formative interviews with inductive thematic analysis [8]. We grouped interviewees' quotes into a set of themes, which became our four key principles for the task of matching sound to video. These principles guide the development of Soundify.

Principle 1: Surface

First, Soundify needs to help editors surface relevant audio clips based on the video content. Most of the editors we interviewed frequently use libraries of high-quality sounds, such as Epidemic Sound, in their sound editing workflows. While it may not seem like much effort to find a sound with keyword searching and add it to a video, the effort piles up with many full-length videos containing hundreds or thousands of video clips and numerous sounds per clip. In addition, editors often see finding and adding background sounds to video as a non-creative task: "I just want it to sound like a forest… it's just a task I want to do along the way. (I3)" Surfacing relevant sounds could potentially speed up an editor's workflow considerably.

Principle 2: Synchronize

Second, Soundify needs to help editors synchronize the surfaced audio clips based on the video clip. For example, given the sound effect of a bicycle peddling, the audio needs to come in when the bicycle appears on the scene and go away when the bicycle exits the scene. Nonetheless, given an audio clip, it can be tedious to manually align it to select video frames. Editors

expressed that synchronizing sound can be a "laborious process (I10)" since it involves a lot of trial and error and suggested that "a lot can be done in automated syncing (I10)".

Principle 3: Spatial

Third, Soundify needs to help editors dynamically convert the audio clip to spatial sound. Using the bicycle example, as the bicycle peddles from left to right, the audio clip should gradually move from the audience's left ear to the right ear (i.e., audio panning). If the bicycle starts off far away and moves closer to the viewer, the audio clip should gradually become louder over time (i.e., audio volume). However, similar to synchronizing, tuning pan and volume parameters of an audio frame by frame can become "the tedious part that no one wants to do (I5)". Editors hoped there could be "a clever way of generating stereo (I3)" since it can be "one of the most difficult and time-consuming parts (I3)".

Principle 4: Stack

Fourth, Soundify needs to give editors the ability to stack multiple audio tracks. Editors expressed that their sound editing workflow involves "a lot of blending and layering (I9)". Similar to how choirs consist of multiple singers singing at different ranges to create a sound that is fuller and has more depth, a good soundscape involves the stacking of multiple audio tracks. Audio stacking usually involves a base layer of ambients and several layers of effects. For example, an editor describes his workflow for stacking audio as: "Ambient room tones in one track, voice-over on another track, and six tracks for effects (I1)". An editor further emphasized the importance of having ambient noise even if there are no clear effects to be added: "It's really important to always have ‘something’... even some ambient street sound to make it sound natural… it is always better to have something than nothing (I3)".

3.1.5 System Implementation

Our four principles are manifested in Soundify and guide its implementation. We first give an overview of our system (Figure 3.2). To reduce the number of distinct sounds to classify across a long video, we split a given video into scenes. We use a boundary detection algorithm based on

color histogram distances [20]. A large distance between the histograms of neighboring frames indicates a scene change. For each scene, we classify for multiple sound effects and an ambient. For each classified effect, we sync the audio clip to the interval in which the corresponding object appears in the video. The classified ambient is used for the entire scene. For each synchronized interval, we mix the pan and volume parameters of the sound effects over time based on the object's position and size in the video. Finally, we stack our matched effects and ambient to produce the final result. In the following sections, we describe in more detail the implementation of the various components including (1) Classify, (2) Sync, and (3) Mix.

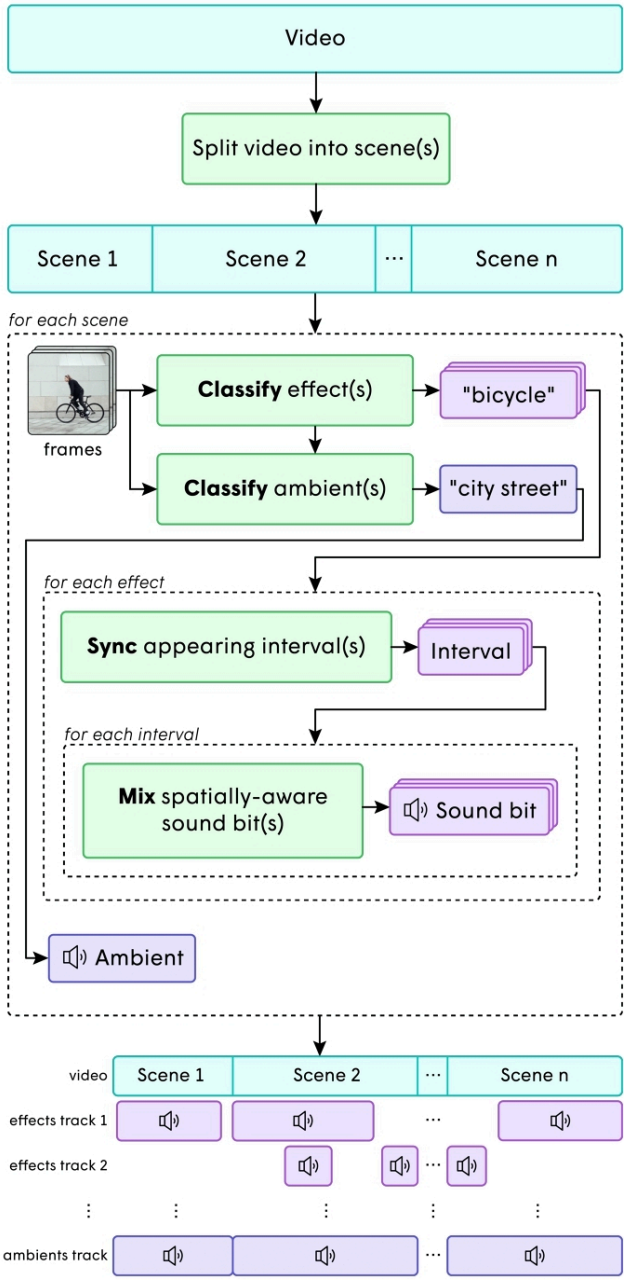

Figure 3.2: Overview of Soundify. Soundify first splits a video into scenes. For each scene, Soundify classifies for effects and ambients. The matched ambient is used for the entire scene. For each matched effect, Soundify performs more fine-grained synchronization by identifying their appearing intervals. For each interval, Soundify mixes spatial sound chunks with computed pan and gain parameters. The final result consists of one or more effects tracks and an ambients track.

Classify

The first stage of Soundify is classification. We match sound effects to a video by classifying "sound emitters" within the video (Figure 3.3). A sound emitter is simply an object or environment that produces sound and is defined based on the sound categories of Epidemic Sound [3], a curated database of over 90,000 high-quality sound effects. To construct a realistic soundscape, we classify each scene for two types of sounds: effects (e.g., bicycle, camera, keyboard) and ambients (e.g., street, room, cafe) (Principle 1). For a given scene, we run each video frame through the CLIP image encoder and concatenate the encoded frames into a vector representation for the entire scene. For each effects label in the sound database, we run it through the CLIP text encoder to return a vector representation for the label. We then perform pairwise comparisons between the encoded scene vector and each encoded effects label vector with cosine similarities and obtain the top-5 matching effects labels for the scene. The user may then select one or more recommended effects (Principle 4). For ambients type labels, we perform the same encoding and pairwise comparison steps. Ambients classification can be more error-prone due to the background potentially being visually out of focus. Thus, we additionally run both the predicted ambients and the previously user-selected effect(s) through CLIP text encoders, and rerank the predicted ambients based on their cosine similarities (Figure 3.4). For example, forest may be ranked higher than cafe if the user had previously selected waterfall as the effect. The user may then select a recommended ambient.

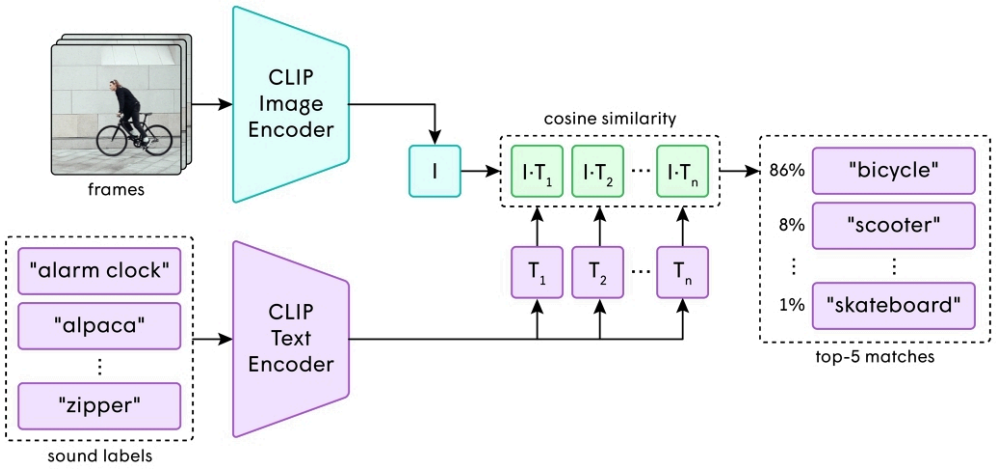

Figure 3.3: Effects classification. Given the frames of a scene and a database of sound labels, Soundify performs pairwise comparisons to predict the top-5 matching sounds.

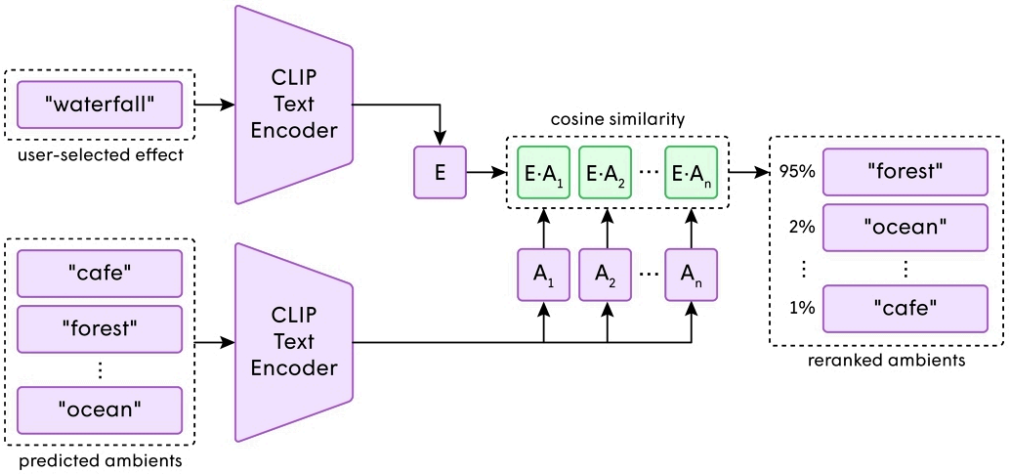

Figure 3.4: Ambients classification. Since ambients classification can be more error-prone, given the user-select effects label and predicted ambients labels, Soundify performs pairwise comparisons to rerank the ambients.

Sync

A sound emitter may appear on screen for only a subset of the scene. Therefore, we want to synchronize effects to when their sound emitter appears (Principle 2) (Figure 3.5). We pinpoint such intervals by comparing the effects label with each frame of the scene. If a sequence of consecutive frames have similarity scores above a threshold, we identify it as an interval. There may be multiple intervals in each scene, such as when a sound emitter disappears then reappears.

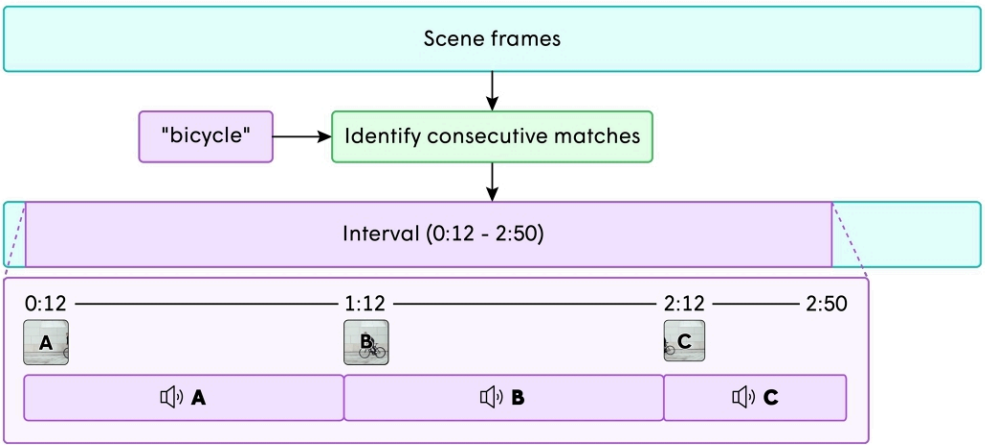

Figure 3.5: Sync. Given the frames of a scene and a sound label, Soundify identifies appearing intervals. An interval is split into chunks. Each chunk takes the first frame as its reference frame.

Mix

Video editors adjust sound according to the state of the scene. For instance, as a bicycle peddles from one side to another, we hear a shift in stereo panning (i.e., sound moves from left to right). As an airplane glides up close, we experience a gain in sound intensity (i.e., sound volume changes). Similarly, we mix an effect's pan and gain parameters over time (Principle 3) (Figure 3.6). To achieve this, we split an effects interval into around one-second chunks (Figure 3.5), mix the pan and gain parameters for each chunk (Figure 3.7), and stitch the chunks smoothly with crossfades. A one-second chunk uses the first image frame as the reference image. We run the reference image through Grad-CAM [32] on the ReLU activation of the last visual layer (ResNet-50 architecture) to generate an activation map (example visualizations of activation maps shown in Figure 3.6). This localizes the sound emitter, allowing the system to take on the capabilities of an open-vocabulary object detector (i.e., works on arbitrary objects). We then compute the pan parameter by the x-axis of the localized sound emitter's center of mass and the gain parameter by its normalized area. Next, we retrieve the effect's corresponding .wav audio file and remix its pan and gain. We prioritize retrieving audio clips that have a duration longer or equal to the occurrence of a sound object on screen. For ambients, we assume a constant environment for each scene. Thus, we retrieve the corresponding .wav audio file and use it across the entire scene with a -5 dB volume adjustment as to not overpower the main sound effects. Finally, we stack all selected audio tracks of effects and ambients for all scenes into one final audio track for the video (Principle 4) (Figure 3.2).

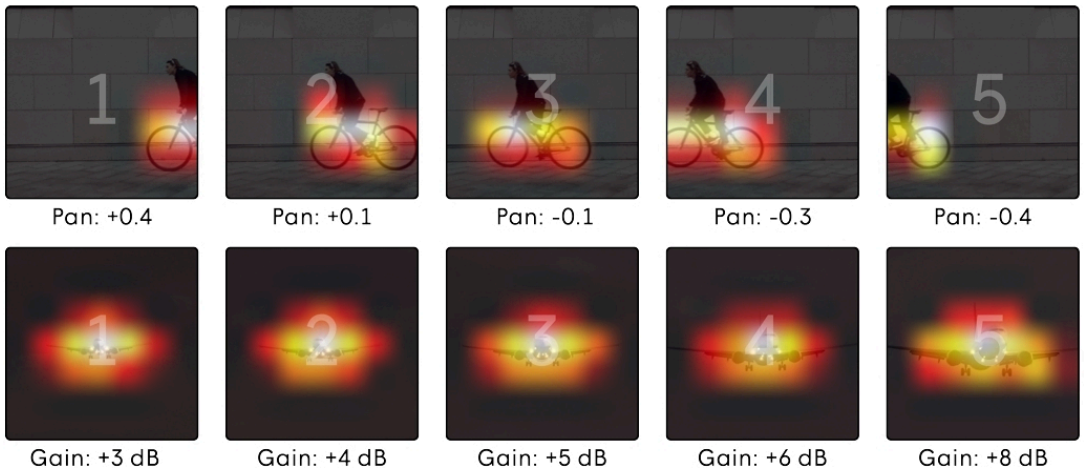

Figure 3.6: Soundify adapts pan (top row) and gain (bottom row) parameters over time based on the heatmap's position and size.

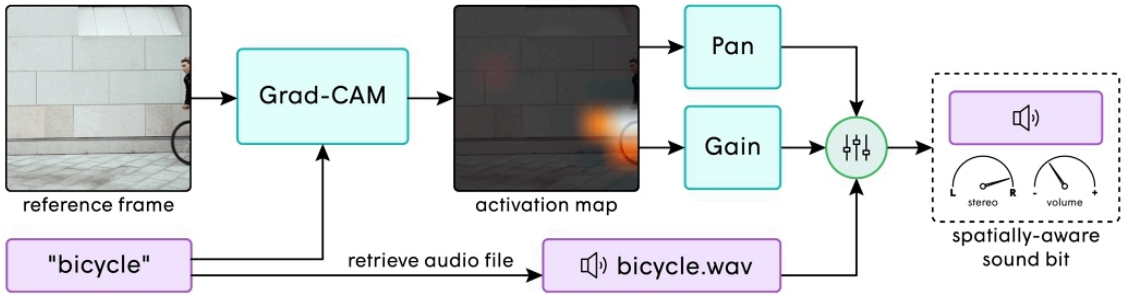

Figure 3.7: Mix. Given a reference frame and a sound label, Soundify retrieves the relevant audio file and mixes its pan and gain parameters, by referencing the activation map, to generate a spatial sound chunk.

Interface

We provide users with an interface to view the system's predictions and make creative sound design decisions (Figure 3.8). We first split a video into various scenes and allow users to adjust the sound effects and ambients for each scene. The user may switch between scenes using a slider (Figure 3.8a). In the sound effects panel (Figure 3.8b), the highest-scoring sound effect is pre-populated by default. The user may add or remove sound effects and stack multiple sound effects (Principle 4) from a multi-selection dropdown menu with sound effects sorted in descending order of their predicted scores. The user may preview the audio file of each selected sound effect. In the ambients panel (Figure 3.8c), the highest-scoring ambient is pre-populated by default. The user may switch to another ambient from a dropdown menu with ambients

sorted in descending order of their predicted scores. The user may also preview the audio file of the selected ambient. After the user clicks on the "Generate" button, we visualize the heatmap predictions in one-second intervals (Figure 3.8d). After the generation is complete, we also allow users to export the video and audio tracks of the video split and numbered by scene (i.e., scene 1 video track, scene 1 audio track, …) as a .zip file so that they may import and use the audio clips in a video editor of their choice.

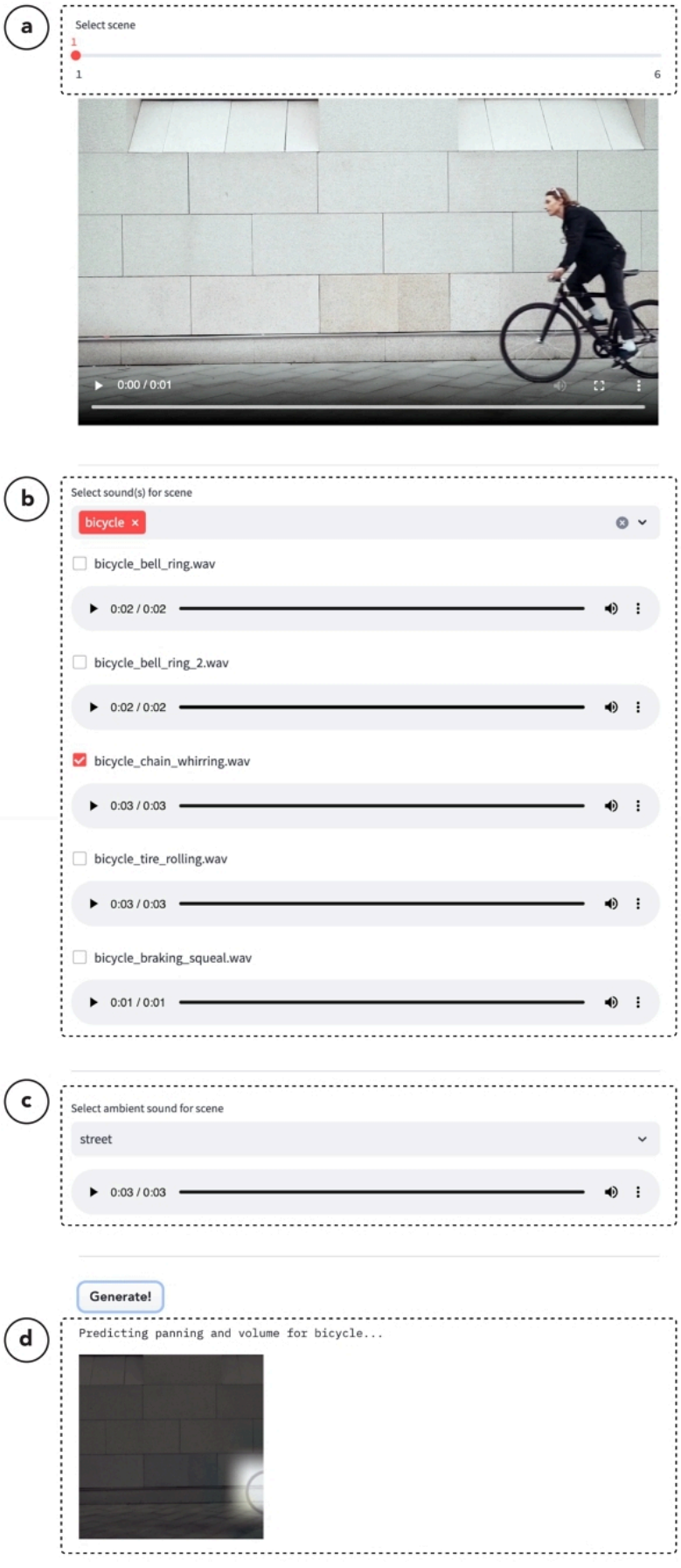
a
Select scene
1
1
6
0:00 / 0:01
b
Select sound(s) for scene
bicycle ×
bicycle_bell_ring.wav
0:02 / 0:02
bicycle_bell_ring_2.wav
0:02 / 0:02
bicycle_chain_whirring.wav
0:03 / 0:03
bicycle_tire_rolling.wav
0:03 / 0:03
bicycle_braking_squeal.wav
0:01 / 0:01
c
Select ambient sound for scene
street
0:03 / 0:03
Generate!
d
Predicting panning and volume for bicycle...

Figure 3.8: Soundify interface. A video is split into scenes and the user may switch between scenes via a slider (a). For each scene, the user may choose one or multiple recommended sound effects (b) and a recommended ambient (c). After the user hits the "Generate" button, Soundify locates the sounds in the video and the interface displays the predictions for pan and gain levels in one-second intervals (d).

3.1.6 Human Evaluation

To test the effectiveness of Soundify in detecting sound emitters and matching sounds to them, we run a human evaluation study on a collection of videos with sounds matched by Soundify against a baseline method built on YOLO [5], a state-of-the-art object detector trained on 328,000 images of 91 categories of everyday objects and humans, many of which can be found in the videos we are testing. YOLO provides a bounding box around objects that can allow for us to track them across frames and adjust pan and volume based on position and size. We do not compare against an audio synthesis model as our baseline as the focus of this paper is on investigating the utilization of existing sound effects clips. The following outlines our experimental setup, procedure, and results.

Audio Clip Collection. For our audio clips, we use a combination of the ESC-50 dataset [27] and a subset of the UrbanSound8K dataset [31]. In total, we obtain a diverse audio collection of 54 categories with 40 audio clips per category, yielding a set of 2,160 audio clips.

Video Clip Collection. We collect video clips from Getty Images, a large database of professional video footage. We query for videos on Getty Images using the audio category labels as keywords and collect the first 80 results for each keyword, while manually filtering out irrelevant videos. In total, we result in a set of 1,105 videos clips that represent diverse and complex scenes containing multiple objects.

Soundify and Baseline Setup

We match audio clips to video, including synchronization and spatial tuning of pan and volume, with two systems: Soundify and the baseline system. The sound matching processes were done automatically by the two systems.

Soundify Setup. We match sound to video with Soundify as detailed in Section 5.

Baseline Setup. Our baseline system is based on YOLO [5], an object detector that draws bounding boxes around detected objects. We use a YOLO model trained on a large variety of categories in the COCO dataset [24]. After the model detects the objects, we take the average x-value of the bounding box coordinates to set the sound panning and the area of the boundary box for the sound volume.

Procedure

We run a Mechanical Turk study to evaluate the videos with sound matched by Soundify and the baseline system. We set the worker qualification requirements of above 95% approval rate and greater than 100 HITs approved. We also ask that workers have headphones with left and right channels to participate in our task. For each video matched with sound, we ask five different people to evaluate it. It takes around 12 seconds to complete a HIT and we pay $0.04 USD per HIT, which is above US minimum wage. The worker does not know whether a given video is matched by Soundify or the baseline system. We ask workers to answer how they feel about each of the following statements on a scale of 1 to 5 (strongly disagree, disagree, neutral, agree, strongly agree):

- The audio matches the type of object shown in the video.
- The audio aligns in time well with the video.
- The volume of the audio matches well with the video.
- The panning of the audio matches well with the video.
- Overall, the audio matches well with the video.
- After collecting the responses, we filter out responses where the worker does not play the video or does not take a long enough time to complete the task.

Results

Figure 3.9 shows an overview of the human evaluation results comparing Soundify and the baseline system. We analyze the results for statistical significance through an unpaired t-test

with Bonferroni correction (5 tests, significance level at $\alpha < 0.01/5=0.002$). Participants report a significantly higher rating for the results generated with Soundify (mean=4.11, SD=0.29) compared to the baseline (mean=3.09, SD=0.69) ($t(1136.1)=-40.80$, $p<0.002$, $r=0.77$, $ds=1.92$) (5-point Likert scale).

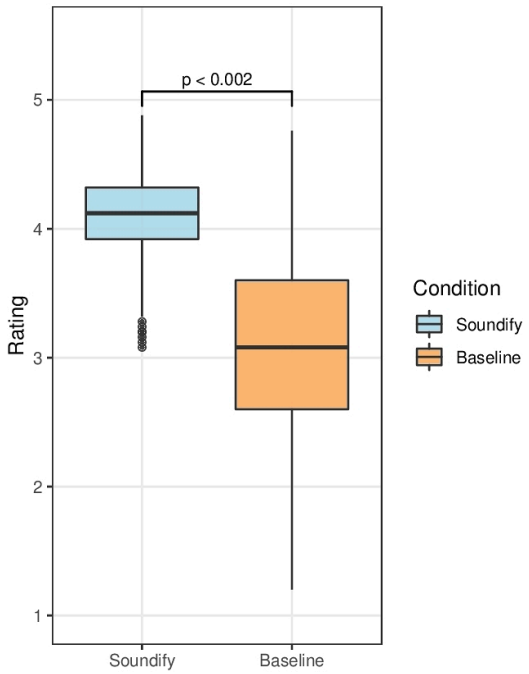

Figure 3.9: Human evaluation results (N=889) (5-point Likert scale, higher is better).

Our results suggest how our extension of CLIP with activation maps can enable open-vocabulary object detection, meaning it can detect objects beyond its own training set and offers greater flexibility to fine-grained audios. In addition, CLIP's heatmap may offer a better approximation of object size and position than YOLO's bounding boxes.

3.1.7 Expert Study

To evaluate the usefulness of Soundify in assisting video editors, we conduct a within-subjects expert study comparing Soundify to a baseline task of manual editing. The following outlines our study design, participants, procedure, and results. Our main research questions are:

1. How would the level of workload for participants be affected with the use of Soundify?

2. How do participants' find the usability of Soundify?
3. How would task competition time change with the use of Soundify?
4. Qualitatively, what would participants see as the pros and cons of Soundify?

Study Design

Independent Variable. The independent variable of the study is the system: Soundify versus the baseline.

Dependent Variable. The dependent variables of the study are workload (RQ1) measured by the mental, effort, and frustration components of the NASA TLX questionnaire [21], usability (RQ2) measured by the System Usability Scale (SUS) questionnaire [9], and task completion time (RQ3) reported by the participant (in seconds). All questionnaire questions are represented on a 7-point Likert scale.

Participants

We recruit 12 professional video editors (10 male, 2 female) aged 24 to 59 (mean=34.17, SD=9.81) from Upwork, a platform for hiring freelancers [4]. We conduct a background survey with the participants before each study to assess their video and sound editing experience. Overall, participants have high self-rated familiarity with video and sound editing (mean=6.58, SD=0.67) (7-point Likert scale) and have several years of experience editing sound for videos (mean=10.00, SD=7.21). All participants regularly use Adobe Premiere Pro for video and sound editing. In addition, participants also have experience with other video and sound editing software such as Adobe After Effects, Final Cut Pro, DaVinci Resolve, Adobe Audition, Avid Pro Tools, Reaper, and Ableton Live. Furthermore, all participants make use of sound effects libraries in their workflows.

Procedure

We conduct the expert study remotely. After receiving the participant's consent, we collect information about individual backgrounds. We then ask the participant to match sound to a video with Soundify and to match sound to another video with the baseline of manually

searching for sounds and adding them in Adobe Premiere Pro. In the experimental condition, the participant may optionally export the results from Soundify and perform further manual editing. We counterbalance both the order of the conditions and the order of the videos. The two videos are of comparable difficulty and contain multiple complex scenes with scenes containing multiple objects. We also ask participants to record the time they spend in each condition. After each condition, we ask participants to complete the NASA TLX, SUS, and task completion time questionnaires. After the participant completes both conditions, we ask the participant to answer open-ended questions regarding the overall experience of using Soundify. The study lasts for approximately 40 minutes. We compensate participants $30 USD for their time.

3.1.8 Results and Discussion

For quantitative analysis, we analyze the scores for workload, usability, and task completion using a paired t-test comparing Soundify with the baseline condition. Figure 3.10 shows an overview of the quantitative results comparing Soundify against the manual editing baseline. For qualitative analysis, we analyze the participants' open-ended responses with deductive thematic analysis [8] according to the dimensions of the quantitative measurements (i.e., workload, task completion time, and usability). The following presents and discusses the results of the expert study.

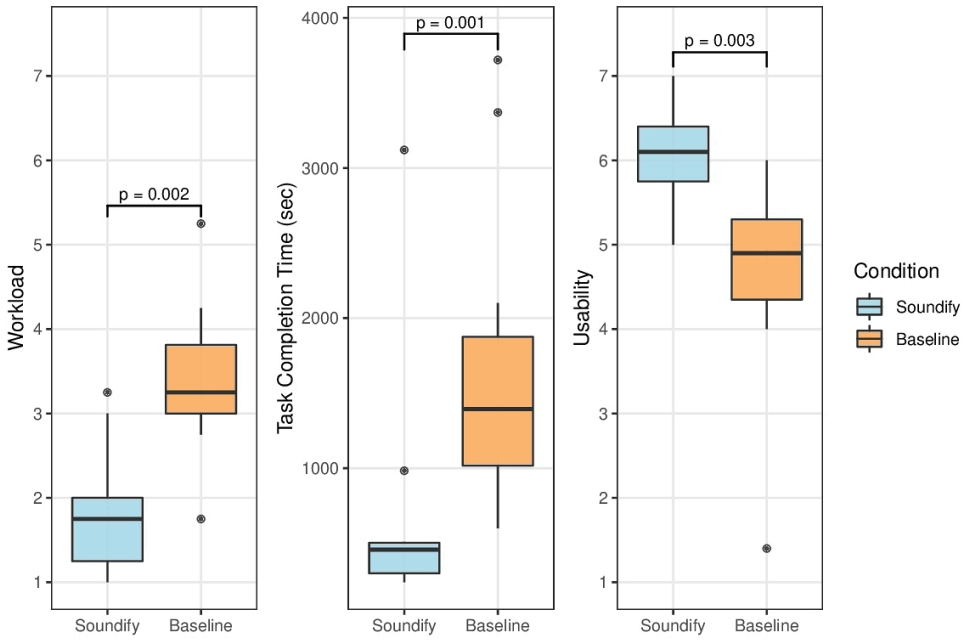

Figure 3.10: Expert study results (N=12). Boxplots from left to right: (a) workload measured with NASA TLX [21] (7-point Likert scale, lower is better), (b) task completion time (seconds, lower is better), and (c) usability measured with SUS [9] (7-point Likert scale, higher is better).

Workload. The differences in workload per participant across conditions pass the Shapiro-Wilk test of normality (W=0.95, p=0.71). We thus compare the differences in workload through a parametric paired t-test. Participants report a significantly lower workload when using Soundify (mean=1.85, SD=0.74) compared to the baseline (mean=3.38, SD=0.87) (t(11)=4.02, p=0.002, r=0.77, ds=1.16) (7-point Likert scale, lower is better) (Figure 3.10a). Participants express that they "do edits that feature a lot of tedious sound effect placement constantly (P5)" and that it's one of their "least favorite parts of the editing process (P5)": "I feel like most sound effect placement in video editing is essentially grunt work. It doesn't take any kind of skill per se. It's just something you have to do, and it's tedious to find and place these sounds (P5)". Participants feel that Soundify "picked appropriate SFX (P9)" and helped "minimize having to work on an unfulfilling part of the job (P5)": "The AI worked like a charm… everything was with such little effort (P6)". Participants enjoy "how little [editing] work was actually needed to be done by the editor (P10)" and being able to feel more like "directing and supervising (P10)" a project: "Any time I can work in dropdown menus rather than a timeline, I'd prefer it. (P5)"

Task Completion Time. The differences in task completion time per participant across conditions pass the Shapiro-Wilk test of normality (W=0.90, p=0.14). We thus compare the differences in task completion time through a parametric paired t-test. Participants report a significantly lower task completion time in seconds when using Soundify (mean=670, SD=795) compared to the baseline (mean=1681, SD=979) (t(11)=4.40, p=0.001, r=0.80, ds=1.27) (time taken in seconds, lower is better) (Figure 3.10b). Participants enjoy "having a tool that creates a base layer of audio (P4)" and being able to "achieve a decent result very quickly (P12)": "I love the instantaneous result of being able to instantly bring a scene to life with sound. (P6)" Participants state that Soundify helps cut down time ("my time spent was cut in half (P10)") and could be especially useful when they have a "high-volume of projects (P3)": "I have so many projects to edit and I went from 35 minutes to 7 minutes… I could be so much more efficient in my editing projects if I were to use Soundify in my workflow. (P8)"

Usability. The differences in usability per participant across conditions pass the Shapiro-Wilk test of normality (W=0.90, p=0.15). We thus compare the differences in usability through a parametric paired t-test. Participants report a significantly higher usability when using Soundify (mean=6.03, SD=0.53) compared to the baseline (mean=4.70, SD=1.23) (t(11)=-3.77, p=0.003, r=0.75, ds=1.09) (7-point Likert scale, higher is better) (Figure 3.10c). Participants find Soundify to be "easy to learn (P3)" and "easy to use with no instructions (P9)": "It was easy to use and accomplished its prescribed goal well. (P1)" Participants enjoy "being able to scroll through the portions of the video [scene-by-scene] (P7)": "I would use the tool [Soundify] just for the ability to split scenes and file management. (P11)" Participants mention that the export function is useful: "I love being about to adjust the audio levels afterward. (P6)" Participants like how the "downloaded files are already cut per scene (P11)" and find that the "saved numbered format [per scene] is very helpful (P11)".

Participants also comment on specific technical components of Soundify (classify, sync, and mix):

Classify. Participants state that Soundify "knocks down the amount of time spent searching for sounds (P9)": "I love that it finds the sound effects based on the visuals so you don't have to go keyword treasure hunting for the right sound (P5)".

Sync. Participants "appreciate the accuracy with which the sound synchronized to each shot (P6)": "I like that it puts the sound in the right place in the timeline for you. (P5)"

Mix. Participants feel that Soundify "predicted the panning well (P7)" and helped "eliminate the inefficient keyframing within Adobe [Premiere] (P7)".

3.1.9 Limitations and Future Work

While Soundify was positively received in our user studies, there are several avenues for improvement that we plan to address for future work. First, we could improve very fine-grained synchronizations for certain sounds, such as footsteps. Currently, Soundify does not support matching footsteps to the exact moments in which the foot impacts the ground. Several potential approaches for exploration may include incorporating motion cues, conducting state analysis, and leveraging the metadata of videos (e.g., timecoded script with sound annotations). Second, CLIP (the base model that Soundify is built on) may occasionally encounter mistakes. One example is CLIP incorrectly classifying an AC repair handyman holding a screwdriver as a person brushing their teeth. To ensure the accurate classification of sound objects at the beginning of the Soundify pipeline, we will continually update Soundify with the latest improved CLIP model [2]. In addition, a potential avenue for exploration may be to finetune CLIP on a large set of labels found in the audio library. Third, video editors also suggested several manual finetuning capabilities, such as custom fades and effects and adjusting EQ (i.e., boosting or damping certain frequencies). We could allow the tuning of additional effects in future work by integrating Soundify into a more comprehensive sound editing software.

3.1.10 Summary

Soundify maps *object-centric interaction* with *class activation maps* in AI models, enabling video editors to treat sound sources as individually controllable objects rather than unstructured pixels on the screen and keyframes on the timeline. Soundify primarily improves the *contextual* characteristic of AI controllability (context-blind → context-aware) and shifts upwards in *intuitiveness* in the intuitiveness-controllability spectrum. From object-level context, the next chapter zooms out to corpus-level context - helping video editors gain an overview of large video collections and navigate thousands of video frames through a map interface.

3.2 VideoMap: Map as an Interface

This chapter was adapted from my published paper: Lin, D. C. E., Caba Heilbron, F., Lee, J. Y., Wang, O., & Martelaro, N. (2024, June). VideoMap: Supporting Video Exploration, Brainstorming, and Prototyping in the Latent Space. In *Proceedings of the 16th Conference on Creativity & Cognition* (pp. 311-327).

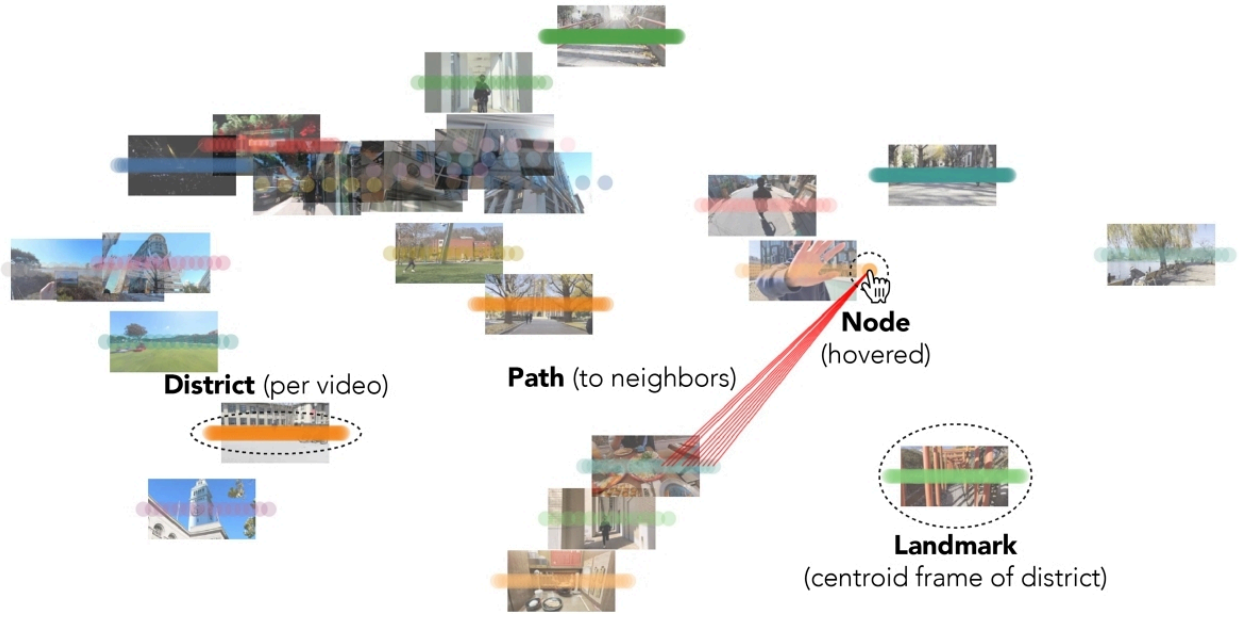

Figure X: VideoMap is a proof-of-concept video editing interface that operates on video frames projected onto a latent space, enabling users to visually uncover patterns and relationships. We introduce map-inspired navigational elements (node, path, district, landmark) to support users in navigating the map.

3.2.1 Introduction

Video editing is a creative and complex endeavor. Consider a typical workflow: As the editor sifts through large amounts of footage, they must first develop a comprehensive understanding of the material at hand and conceptualize a narrative. The editor then needs to select the individual clips to use and identify connections between them to weave them together with suitable transitions. As the editor refines the edit, they may constantly experiment with different editing ideas.

We observe that popular video editing interfaces used by video editors often feature a sequential editing timeline, with designs rooted in the metaphors of hand cutting and splicing film [5], supported by a grid-like asset management panel, where clips are listed by filename order or time of creation [8]. While these editing interfaces functionally support the task of assembling video frames in a specific order, they do not have explicit mechanisms to support the creative exploratory aspects of editing, such as developing a holistic understanding of all video footage, identifying connections between them, and rapidly experimenting with multiple editing ideas. We believe that there is an opportunity to build a new video editing interface that better supports this creative process. We aim to augment the way editors see and interact with their video clips during the early stages of video editing, including exploration, brainstorming, and prototyping.

In this research, we take inspiration from latent space exploration tools that help users find patterns and connections within complex datasets [49]. Latent space exploration tools leverage data processing and recent machine learning techniques to transform data into vector representations and enable users to visually explore the data on a spatial interface. Building upon this insight, we ask if we can similarly project video frames onto a latent space and allow editors to edit videos within this space.

To investigate this possibility, we developed VideoMap, a proof-of-concept video editing interface based on latent space representations of video frames. In our system, we encode video frames onto latent spaces that are meaningful for video editors and support specific video editing tasks. For example, we created a latent space in which video frames containing similar shapes are near each other to help editors identify opportunities to link together scenes with similar shapes and create seamless transitions. Editors may switch between different types of latent space maps that encode information meaningful to video to focus on different types of tasks, a concept we call “swappable lenses.” To facilitate more intuitive navigation of the map, we introduce a set of map-inspired navigational elements: nodes, districts, landmarks, and paths [51]. We then designed three components for VideoMap that utilize the latent space map

to support three common video tasks. Specifically, we designed 1) Project Panel to help editors explore video footage, 2) Paths Explorer to help editors find suitable video transitions, and 3) Route Planner to help editors quickly prototype rough cuts and try out different editing ideas.

In a user study, we invited both professional and non-professional video editors to test VideoMap. We found that video editors were able to effectively use VideoMap's components to perform the editing tasks listed above. Editors expressed that VideoMap provides a user-friendly editing experience, reduces tedious grunt work, enhances the overview capability of video footage, helps identify suitable video transitions, and enables a more exploratory approach to video editing. We further explored the design space of VideoMap by showcasing how VideoMap can be customized and extended to support additional applications, including summarizing videos, finding highlight moments within videos, and text-based video editing.

This research thus makes the following contributions:

- We introduce VideoMap, a proof-of-concept video editing interface that operates on video frames projected onto a latent space. We support intuitive navigation of the latent space through map-inspired navigational elements and facilitate transitioning between different latent spaces using swappable lenses.
- We built three VideoMap components to support editors in three common video tasks: exploring video footage, finding suitable video transitions, and quickly prototyping rough cuts.
- We demonstrate the effectiveness of VideoMap in supporting video editors complete creative editing tasks through a user study (N=14). Editors felt that VideoMap provides a user-friendly editing experience, reduces tedious grunt work, enhances the overview capability of video footage, promotes editing continuity, and enables a more exploratory approach to video editing.
- We further demonstrate the versatility of VideoMap by implementing three extended applications, showcasing how future developers may customize and extend VideoMap for additional use cases.

3.2.2 Related Work

Our work is situated among literature on video editing tools, video browsing interfaces, and latent space exploration tools.

Video Editing Tools

VideoMap contributes to extensive literature on enhanced video editing tools. In this section, we discuss some past techniques, such as leveraging video metadata, pen input, tangibles, text-based editing, and automation.

Early research on video editing tools explored how data embedded within and associated with videos could be used to create new editing interfaces. One of the seminal works, SILVER [14], uses video metadata to support editors with additional panels to interface with and edit video footage, such as storyboards, editable transcripts, and timeline views. SILVER shows how the use of video metadata allows for smart editing capabilities beyond simple cutting, trimming, and arranging of video frames on the timeline. [27] analyzes video content for fast motion or zooming and computes a “suitability” score for each video frame to help editors decide which clips to include in the edit. This suitability score takes information about the video embedded in the pixels and makes it available as metadata for editing purposes, showing how metadata derived computationally from the frames can be leveraged to support creative editing work. VideoMap builds upon these works and leverages new kinds of metadata derivable with modern machine learning techniques.

Early researchers have also looked into alternative ways to arrange and trim videos. [15] proposes a way to edit video where the user gives voice comments over the video and the system creates a newly edited video based on the comments. Other research has explored augmenting video editing using pen-based technology, allowing editors to edit videos with direct manipulation [12, 13, 69]. Another thread of research explores using tangible interfaces to allow co-located users to edit videos collaboratively [54, 67, 80].

More recent research have mostly explored developing video editing tools that cater to a specific domain. A large thread of work focuses on building editing tools for text-focused videos, such as voiceover videos and talking head videos. [41] helps users edit dialogue-driven scenes by matching video clips to relevant dialogue. [26] lets users edit talking-head videos by editing a text transcript. [35] helps recommend b-roll video clips via interactive transcripts. Similarly, [74] lets users edit text and automatically recommends video clips they filmed to use in the edit. [42] helps users create videos by recommending matching images over text and transcripts through the notion of word concreteness (i.e., the extent to which a word describes something that can be visually experienced). [76] allows users to add images to podcasts by using natural language processing to identify important geographic locations and descriptive keywords. Today, several commercial, text-based video editing tools allow users to edit videos via a paired transcript, including Descript [1] and Type Studio [6]. In Section 7.3, we demonstrate how VideoMap may also be extended to support text-based video editing.

There have also been efforts to automate the video editing process using algorithmic techniques, including identifying highlight segments [32, 46, 75], mimicking professionally-created videos [18, 33, 34, 56], instructing editing objectives in natural language to an LLM-powered agent [72], and automatically synthesizing sound based on video content [45]. In addition, researchers have also explored automatically converting various forms of content into videos, such as web page to video [21], markdown to video [20], and physical demonstrations to video [22]. Although the main objective of VideoMap is not on automatic video editing, we designed a feature that supports editors in automatically generating an initial draft of a video (see Section 4.4), suggesting its potential to support new kinds of video editing automation.

Video Browsing Interfaces

A related research area to video editing tools focuses on interfaces to support video browsing. Browsing and selecting videos from a large set of raw footage is an important first step in the video editing process. A large body of research investigates the use of direct manipulation

techniques [36] for video browsing. Many works have explored video playback by allowing the user to directly click and drag objects within the video, such as clicking on a car and dragging it along the road to play the video over time [28, 38, 40, 55]. [61] creates a timeline with dynamic playback speeds to emphasize important content. Another thread of research supports users in browsing through videos by revealing additional information, such as additional thumbnails [37, 52], visualizing various types of metadata such as brightness intensity and hue histograms [70], enhancing the playback of software tutorial videos by leveraging crowdsourced data from previous user interactions with the tutorials, [78], and improving the playback of surgical videos by surfacing related surgical videos that perform similar surgical steps [39]. In VideoMap, we leverage the content embedded within videos to enable enhanced browsing of video clips and visualization of connections between video clips. This allows editors to explore videos based on their content, rather than merely viewing them as a list of files.

In addition to visual browsing methods, researchers have also explored adopting text-based browsing and navigation for content-heavy videos, such as for lecture videos [58, 77], movies [57], presentation videos [59], and how-to videos [17]. Such text-based browsing methods allow editors to quickly find the right moment in the footage, helping them focus more on composing the video than on searching for content. In VideoMap, we also leverage text-based search of features derived from a semantic latent space to allow editors to find content within large amounts of footage (see Section 4.1.1).

Latent Space Exploration Tools

With advancements in machine learning, researchers in visualization and interpretable machine learning communities have recently explored methods for visualizing the embeddings of neural networks. These embeddings are vector representations of data that have been learned by neural networks. A common initial step of such works involves reducing the dimensions of these vector representations, which are often high-dimensional, to just two dimensions. This allows for the data to be visually plotted using x and y coordinates, resulting in a 2D latent space. Various techniques can be used to reduce the dimension of embeddings while retaining as

much information as possible. Some popular methods include t-SNE [71], HSNE [60], PCA [9], and UMAP [53]. In VideoMap, we adopt dimensionality-reducing preprocessing to develop a new interface for viewing and working with video footage.

Given a 2D latent space, researchers have investigated the development of interactive interfaces to support user exploration [66]. [49] develops an interface to help users interactively discover meaningful relationships among data points in the latent space. [25] allows users to discover structural relationships in data, such as the association of items within groups and the hierarchies of items between groups items. Several works have investigated interfaces for visualizing semantic relationships of text data [48, 79]. [16] analyzes cooking processes at scale by clustering recipes with respect to their structural similarities and facilitates pairwise comparisons between recipes. [24] allows users to visualize a latent space of gestures. [29] support users in visualizing a latent space of human interpretable concepts, such as cars. Many researchers have also worked on tools for comparing multiple latent spaces, such as by highlighting visual changes [65] and through side-by-side comparisons [10, 44]. Most relevant to our work, [73] train video-specific autoencoders (i.e., autoencoder models overfitted to the task of regenerating the videos) to visualize video frames on a latent space, which could be used to perform video manipulations such as inpainting and creating video textures (i.e., looping videos). In VideoMap, we draw inspiration from latent space visualization interfaces to assist video editors in identifying patterns and connections within video footage. We create latent spaces that are meaningful for video editors, representing various aspects of video editing, such as concepts of semantics, color, and shape.

3.2.3 Design Goals

We distill three principles to ground the development of VideoMap based on theories and practices in video editing. These guiding principles include providing an overview of the video footage, maintaining editing continuity with seamless transitions between clips, and facilitating exploration and experimentation throughout the editing process.

Principle 1: Overview

As a time-based medium, video editing often requires editors to spend a considerable amount of time sequentially playing through and reviewing footage on an editing timeline to develop an understanding of the material they are working with. In VideoMap, we aim to overcome the sequential nature of browsing through videos by projecting video clips onto a 2D latent space, which offers a more comprehensive overview of the footage at a glance. Findings from Craft and Cairns [23] suggest the patterns and themes in data may only be seen from a vantage point that comprises the whole view. Map-like interfaces, such as plots on a 2D latent space, can provide such an overview of data and have been shown to allow people to better understand the thematic information across a dataset [31]. We posit that with an enhanced overview, editors will be able to more easily identify patterns and structure and conceptualize a narrative thread from a set of video clips. Furthermore, we designed "swappable lenses" for editors to easily switch between different types of overviews (i.e., different latent spaces) to uncover different types of patterns within the footage, such as patterns in semantics, color, or shape (see Section 4.1.2). We designed the Project Panel component to test how VideoMap can support editors through an enhanced overview capability.

Principle 2: Continuity

Continuity is one of the key principles of video editing. In the 2012 Asia-Pacific Symposium on Creative Post-Production, Richard Crittenden, author of the book Film and Video Editing, comments: "It remains true that good editing tends to be the art that conceals art". The goal of continuity is to connect clips seamlessly so that the edits are "invisible" to the viewer [63]. One editing technique that editors use to achieve continuity is called match cutting [3]. For example, the editor could cut from a video frame of a bagel to a video frame of a donut — using two different objects with similar shapes to transition between present and past scenes, for example. In VideoMap, we aim to assist editors in achieving continuity by recommending seamless video transitions. Since VideoMap operates in the latent space, we suggest video transitions for a video clip or frame by finding near neighbors in the latent space. Taking the shape-based match cut example: By creating a latent space where video frames containing

similar shapes are organized near each other, the bagel video frame and the donut video frame will be near neighbors in the latent space. We designed the Paths Explorer component to test how VideoMap can support editors in achieving continuity in editing.

Principle 3: Exploration

Even with a well-defined narrative and a principle of continuity to follow, video editing remains a process that involves exploration and experimentation. Film director Ridley Scott likened editing to the process of painting: “Editing is like painting, only with film. You are constantly making choices and trying to find the best way to tell the story.” In VideoMap, we aim to provide editors with a wide range of exploration mechanisms. First, we created four map-inspired navigational elements to help editors explore the latent space: nodes, paths, districts, and landmarks [51]. Second, we implemented a semantic searching mechanism in the Project Panel component, allowing editors to quickly locate and discover clips using text prompts. Third, we recommend multiple alternative video transition options for editors in the Paths Explorer component. Finally, we designed the Route Planner component, which can automatically assemble video cuts from user-selected clips, enabling editors to quickly test and preview the results of different editing ideas.

3.2.4 System Implementation

Our three principles are manifested in VideoMap and guide its implementation. The following outlines the implementation of our system. We first detail how we built the base layer latent space map that powers the VideoMap interface. We then go through how we designed VideoMap's main components, including the Project Panel for exploring video footage, Paths Explorer for finding suitable video transitions, and Route Planner for prototyping video route cuts.

Building the Map

We implemented VideoMap's base layer by first creating the 2D latent space. We then enable flexible reconfiguration of different latent spaces using swappable lenses, and support intuitive navigation of the map through map-inspired metaphors.

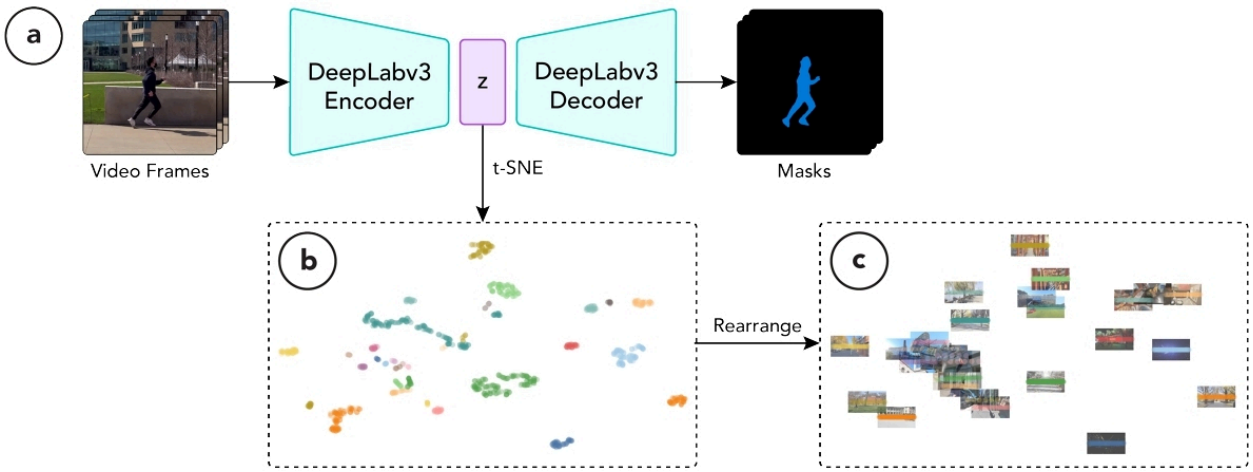

Figure X: The pipeline for creating the shape lens. We pass video frames through DeepLabv3, an image masking model, to extract shape-related vectors (a). We then apply t-SNE to reduce the vectors to two dimensions (b) and visually rearrange the vectors by video to make the videos more skimmable (c).

Creating the Latent Space. VideoMap is based on a latent space representation of video frames. Specifically, we encode video frames into vector representations such that they are meaningful for particular video editing tasks. For example, we can create a latent space map to help editors find shape match cuts, a type of video transition which links together two scenes that contain similar shapes to create a smooth transition (see an example from Kubrick's 2001: A Space Odyssey). To create this map, we pass video frames through DeepLabv3 [19], an image masking model, with a ResNet [30] backbone and extract a 512-dimensional vector representation right after the last visual layer (Figure 2a) Our insight is that due to the mask generation objective, which extracts the dominant shape(s) of an image, vector representations extracted from DeepLabv3 implicitly encode valuable shape-related information. In other words, video frames that contain similar shapes should have more similar vector representations.

Next, we project the vector representations of all video frames into a multi-dimensional latent space. To allow editors to browse through the latent space, we apply t-Distributed Stochastic Neighbor Embedding (t-SNE) [71] to reduce the latent space to two dimensions. In the first iteration of VideoMap, we visualized the vector embeddings directly according to the coordinates generated from the t-SNE (Figure 2 b). However, in our pilot tests, editors felt that this was too messy. They also wished to discern groups of points that belong to the same videos more easily and be able to "play through" the videos. Therefore, in the current iteration of VideoMap, we visually rearrange the points on the map by video (Figure 2 c). We first compute the centroid of all points for each video, then we then visualize frames as points horizontally across the centroid at equal distances. We also display the centroid video frames. Editors can thus play through each video by scrubbing through the points horizontally. From our user study, we found that editors considered this design to be easy to use (see Section 6).

Developing Swappable Lenses. We call the different methods of encoding video frames into meaningful vector representations as different lenses. In Section 4.1.1, we create a shape lens. Under the shape lens, video editors can identify shape-based match cut opportunities by looking at neighbors on the latent space. Similarly, one may develop new lenses by finding other ways of encoding useful vector representations for other use cases. In our proof-of-concept, we implemented a color lens (video frames organized by color) and a semantic lens (video frames organized by meaning). To create the color lens, we first extract 3D histograms from each RGB video frame using 8 bins per channel and normalizing with range = [0, 256], then flatten the histogram to generate a 512-dimensional color-based vector. To create the semantic lens, we encode each video frame through the image encoder of CLIP [62], a neural network that has learned semantic concepts, to generate a 512-dimensional semantics-based vector. Rather than creating a one-size-fits-all map, we allow editors to flexibly swap between different lenses and reconfigure VideoMap according to different criteria. We demonstrate examples of how video editors creatively made use of the different lenses for different editing tasks in Section 5.4.2.

Navigating with Nodes, Paths, Districts, and Landmarks. We designed four map-inspired navigational elements to help editors navigate through the map (Figure 1). The four elements are nodes, paths, districts, and landmarks. These elements are inspired by Lynch's seminal The Image of the City [51], which studied the elements people use to form mental maps of a city.

Nodes. The node is a point (i.e., a vector representation of a video frame) that the user hovers over with their cursor. We follow Schneiderman's details-on-demand mantra [64] by expanding the node's detailed information, such as its corresponding video frame, video filename, and timecode.

Paths. Paths are lines that connect from a user-selected node to its neighboring points on the latent space. When a user clicks on a node, we display ten paths that start from the node and connect to the ten closest neighboring points, which are the points with the smallest cosine distances to the node. We exclude neighboring points that belong to the same video as the node to ensure that the paths represent video transitions. To obtain an accurate representation of neighbors, we calculate the distances based on the original vector representations rather than the rearranged coordinates on the visual map (see Section 4.1.1).

Districts. Districts are groups of points with common characteristics. Throughout most of this paper, we present the version of VideoMap where points that belong to the same video are grouped into a district. Note that paths connect across two different districts (i.e., video transitions). We color-code the points by district to visually separate them. We also consider alternative methods for defining districts, such as through groupings of semantically similar video frames, in Section 7.1.

Landmarks Landmarks are reference markers on the map. With landmarks, the user can obtain an overview of the map at a glance (Principle 1). We visualize the corresponding video frames of some points on the map as landmarks. Throughout most of this paper, we present the version of VideoMap where we generate one landmark for each district by visualizing the video frame

of the point closest to the district's centroid. We also explore the option of allowing users to define custom landmarks in Section 7.2.

Project Panel

The Project Panel is the first component that the editor can use on VideoMap (Figure 3). We designed Project Panel to enable editors to explore video footage more easily (Principle 1). Traditional video editing software typically feature a grid-like interface that displays all of the editor's video files ordered by filename or time of creation. In VideoMap's Project Panel, instead of displaying a standard grid, we rearrange the display of the video files based on the selected lens. For example, Figure 3 displays a collection of videos in the Project Panel under the semantic lens. The editor can swap between different lenses to rearrange the layout and discover new perspectives on the video footage. To reproduce the video thumbnails seen in traditional video editing software, we display the landmark for each video file. The editor may play through the video files by scrubbing the landmarks from left to right. The editor may also zoom in or out of the map by scrolling to view regions of the map with less clutter.

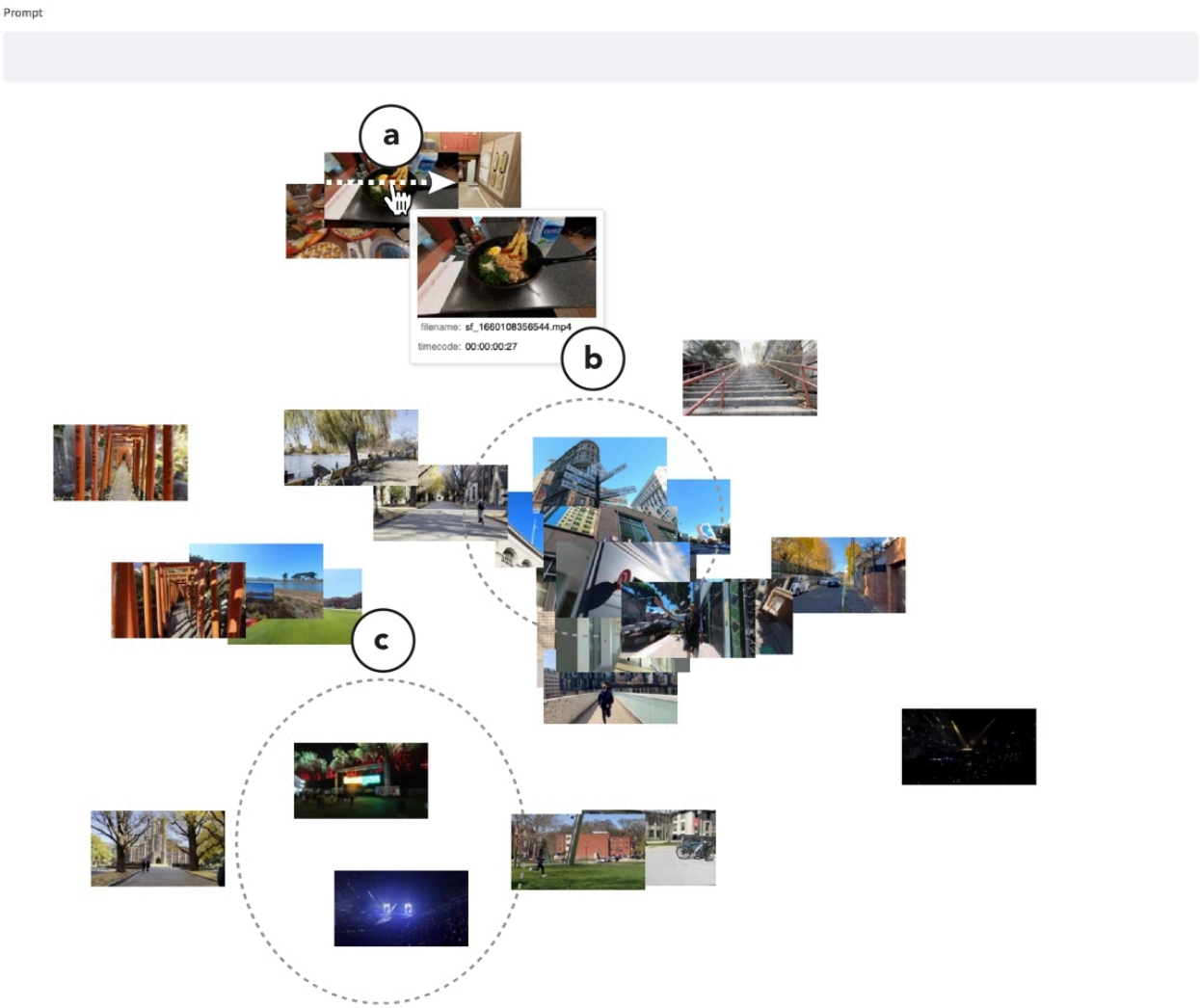

Figure X: VideoMap's Project Panel component. The figure shows a collection of videos organized under the semantic lens. The editor can play through the videos by scrubbing the landmarks from left to right (a). Example semantic clusters of videos are shown in (b) for several videos containing streets and buildings and (c) for concert videos.

Prompts. The editor can filter videos with text prompts (Principle 3) (Figure 4). For example, by typing torii gates, the editor can highlight these videos and fade out the rest. We encode the text prompt using a CLIP text encoder and compare the text embedding to the image embeddings of video frames, encoded by a CLIP image encoder (i.e., the vector representations under the semantic lens). From our user study, we found that editors felt that this method handled prompt searches effectively (see Section 5.4.2).

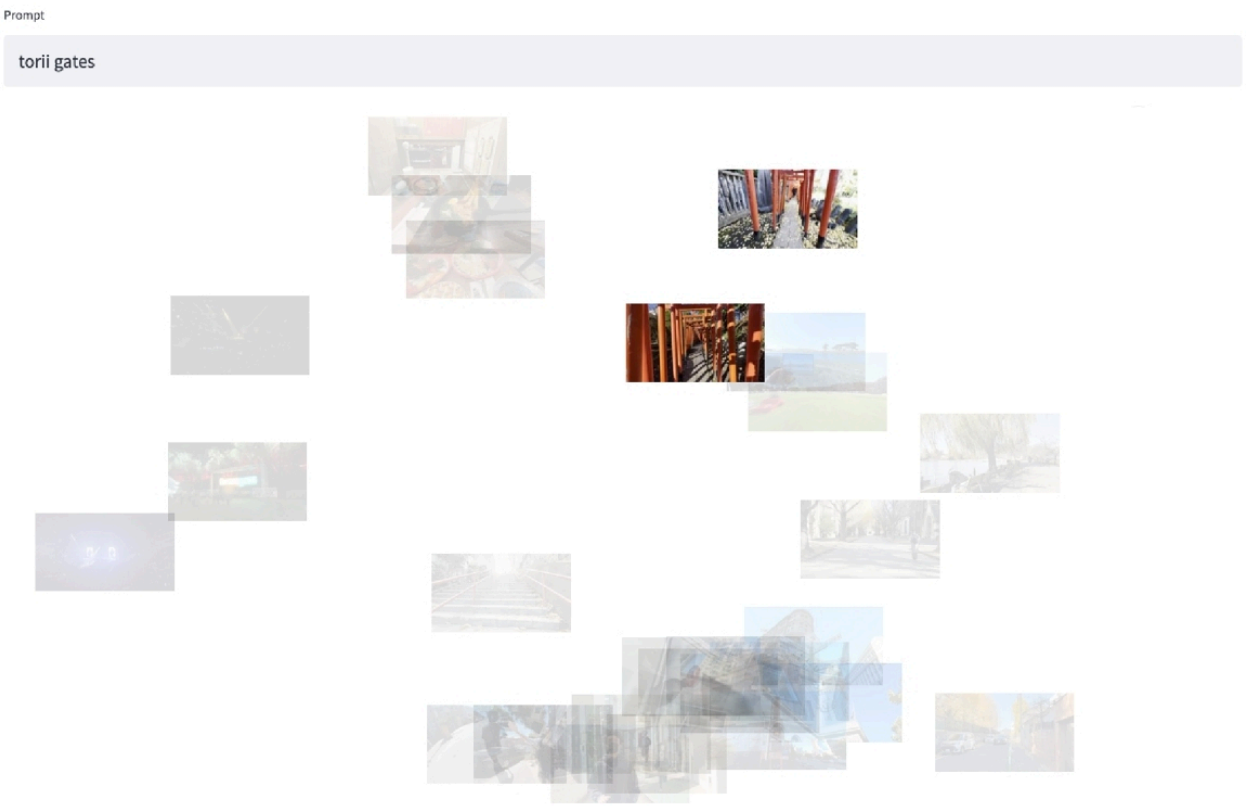

Figure X: The editor can search for videos using text prompts (e.g., torii gates).

Paths Explorer

Paths Explorer is the second component that the editor can use on VideoMap (Figure 5). We designed Paths Explorer to assist editors in finding suitable video transitions (Principle 2). Editors typically arrange videos in a sequential timeline, which can make it difficult for editors to identify patterns of video frames at different times and across different videos. With Paths Explorer, we overcome the inherent sequential nature of videos by arranging them on a 2D map, thereby removing the limitations of viewing video frames along a 1D timeline (Principle 1). While we only display each video's landmark in Project Panel, we display each video's individual video frames as points in Paths Explorer. We color-code the points by video (districts) and fade the landmarks to reduce visual clutter. When the editor clicks on a point, the interface draws 10 paths connecting to nearest neighbor vector representations from other video(s) (Figure 5 b). These paths represent potential video transitions that the system suggests based on the selected lens. For example, Figure 5 displays recommended paths that connect between videos

with similar color composition (having two prominent blocks of color divided at the horizon) under the color lens. The editor can switch between the lenses to discover different types of video transitions.

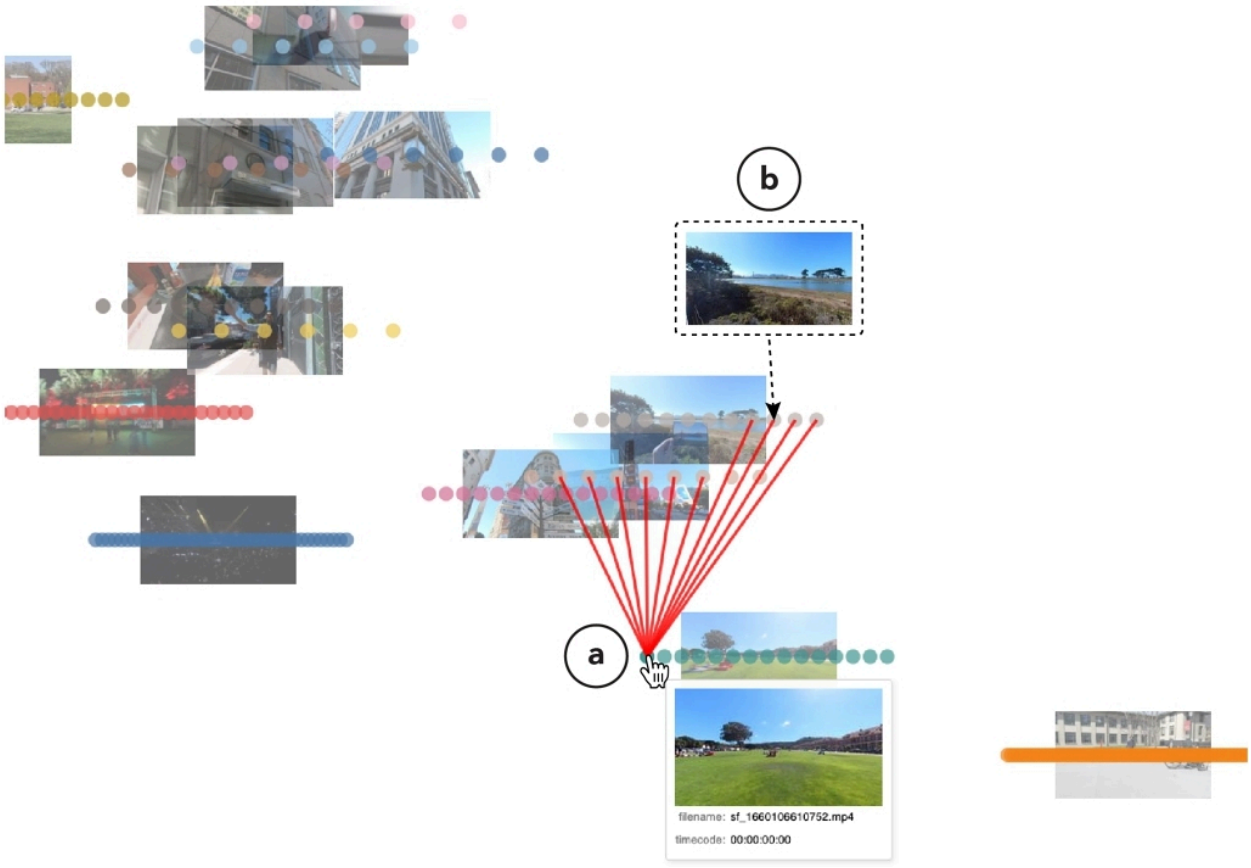

Figure X: VideoMap's Paths Explorer component. The editor can select a video frame to display ten video transition suggestions based on the selected lens (a). For example, a video frame with a similar color composition is recommended under the color lens (b).

Route Planner

Route Planner is the third component that the editor can use on VideoMap (Figure 6. We designed Route Planner to help editors automatically generate rough cuts, which is an initial rudimentary edited version of a video, to quickly explore different video edits (Principle 3). With Route Planner, the editor can select several video clips and the system will automatically find an

optimal route across paths (see Section 4.3) that connect these video clips, based on the selected lens. For example, Figure 6 displays an example route that connects across multiple videos containing the same running action under the shape lens. One can think of this component as being analogous to the Google Maps route planner [2]. The paths connecting video clips are like “streets”. Route Planner finds the fastest route that travels through the streets and passes through each of the selected video clips once. Since the “distances” between video clips on the map are determined based on the selected lens, the editor can switch between the lenses to try out different rough cuts based on different criteria, such as switching to color to connect clips with similar color grading or semantics to connect clips depicting similar objects or activities.

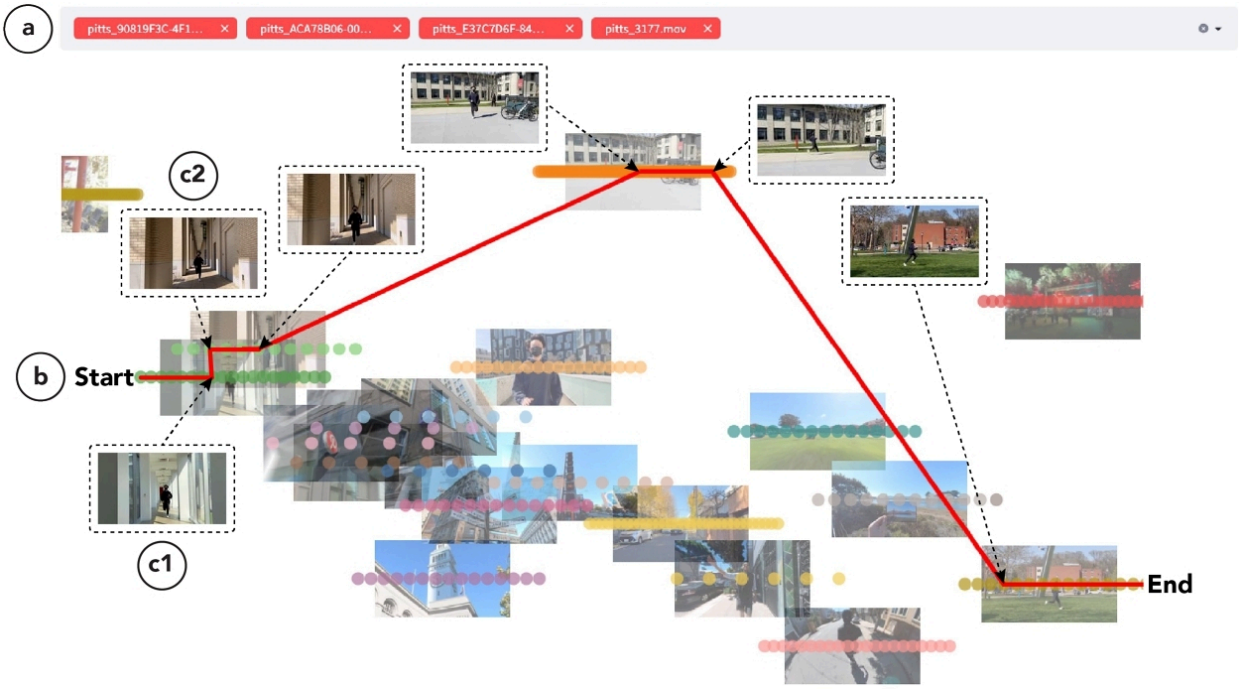

Figure 6: VideoMap's Route Planner component. The editor can select several video clips to automatically generate a rough cut video based on the selected lens (a). For example, the editor can create a video of a person running through various backgrounds under the shape lens (b). Route Planner automatically finds the optimal video transitions (c1 and c2).

To implement Route Planner, we first find the shortest path for all combinations of video pairs (i.e., optimal video transitions). These paths serve as the “streets” that connect across different

videos. Next, we find the shortest route along the streets that passes through all videos (i.e., the Shortest Hamiltonian Path Problem) with a dynamic programming approach. Finally, we visualize the route on the interface and render the rough cut video by traversing through the route. Since the streets are undirected, some videos may be reversed during the traversal. This can lead to interesting results where videos are connected with one playing forward and one playing in reverse.

3.2.5 User Study

We conducted a user study to observe how professional and non-professional video editors might use VideoMap to complete creative editing tasks, including developing an overview of the video clips they have, finding unique transitions and cuts between clips, and crafting a rough cut of a video. We aimed to address three research questions with the study:

1. How can VideoMap assist editors in exploring video footage?
2. How can VideoMap assist editors in brainstorming suitable video transitions?
3. How can VideoMap assist editors in rapidly prototyping rough cuts?

Study Design

To address our three research questions, we asked video editors to perform the three video editing tasks using VideoMap. Specifically, we asked participants to use the Project Panel to explore and familiarize themselves with a collection of video footage, use Paths Explorer to find video transitions, and use Route Planner to create rough cuts. During the study, we asked participants to share their computer screens and perform a think-aloud [43] describing what they were doing and thinking while using VideoMap.

We collected data using three methods. First, we asked participants to complete a questionnaire where participants provided written descriptions on completing the tasks (e.g., a description of how the video clips are organized) as well as feedback on their experience using VideoMap in contrast to how they would perform the same task using a video editing software they are familiar with (see Questionnaire in Supporting Documents). Additionally, participants

attached screenshots of the interface (e.g., a screenshot of a video transition shown in Paths Explorer that they found interesting) and video outputs from completing the tasks (e.g., a rough cut video generated by Route Planner). Second, we recorded the participants' computer screens and transcribed their spoken audio throughout the sessions. Third, the first author silently observed participants working and noted down interesting insights, such as participants' thoughts while using the system, interaction observations from the screen captures, and the creations participants made while completing the tasks. Overall, the data collected allowed us to gain rich insights into the participants' experiences using VideoMap, as well as visual evidence of how VideoMap was used and could support editors in accomplishing fundamental video editing tasks.

Video Collection. We gathered a sample of video clips for participants to use during the editing tasks in our studies. This collection comprises 30 diverse videos, filmed by the authors in various locations including San Francisco, Pittsburgh, and Tokyo. The content covers a variety of categories, including objects like food and architecture, scenery like streets and grassy fields, and actions like running and scootering. The videos were captured using different camera equipment, such as mirrorless cameras, smartphones, and 360 action cameras. The video clip lengths vary from a couple of seconds to several minutes long. Figure 3 provides an overview of the sample video collection displayed in VideoMap's Project Panel.

Participants

We recruited 14 participants (8 male, 6 female) aged from 23 to 55 (μ =28.21, σ =8.78). Seven of the participants are professional video editors recruited from Upwork [7], a platform for hiring creative freelance workers, and seven of the participants are non-professional video editors recruited at our institution via word-of-mouth and snowball sampling. We recruited both professional and non-professional individuals to test VideoMap's ability to support professional editing work as well as make video editing more approachable for non-professionals. We conducted a background survey with the participants before each study to assess their video editing experience. Professional participants have high self-rated familiarity with video editing

(μ =5.86, σ =1.46) (7-point Likert scale) and many years of experience (μ =8.07, σ =2.55). Non-professional participants have moderate self-rated familiarity with video editing (μ =4.43, σ =1.27) and several years of experience (μ =3.00, σ =1.15). Participants use various video editing software such as Adobe Premiere Pro, Final Cut Pro, Movavi Video Suite, iMovie, CapCut, and Google Photos Movie Editor.

Procedure

We conducted the studies over video conference (Zoom). We asked participants to try out the three components of VideoMap (Project Panel, Paths Explorer, and Route Planner) and complete a video editing task using each component. The following details our step-by-step procedures.

After obtaining the participant's consent, we collected background information about their prior video editing experience. Next, the participant tried out VideoMap. First, they used the Project Panel to familiarize themselves with the sample video collection and think about how they could create a video montage using it. We also asked participants to try the different lenses (semantic, color, shape) and try searching for clips by typing different prompts in the prompts box. We asked the participant to write down three prompts that yielded interesting results. Second, the participant used Paths Explorer to find video transition opportunities. We asked the participant to record three transitions that they found interesting during their exploration. Third, the participant used the Route Planner to quickly create rough cuts for a video montage using the provided clips. Participants used the Project Panel to determine which video clips to select in the Route Planner for the rough cut. We asked the participant to record one rough cut that they found interesting. After trying out the three components of VideoMap, we asked the participant to complete a post-study written survey about their experiences. We also asked participants to contrast their experiences with how they would typically perform the same tasks using existing video editing software. The study lasted for approximately 1 hour.

3.2.6 Results

We analyzed our data, including observation notes, audio transcripts, and form responses, with inductive coding to identify common themes. The following presents our findings. We note direct quotes from professional participants as P and non-professionals as N.

Project Panel. Participants were able to use Project Panel to explore the video footage by familiarizing, grouping, and uncovering structure among the clips (RQ1). In addition, participants also noted additional use cases of Project Panel, such as supporting visual ideation and helping with planning new shots to take. As part of the first task, we asked participants to brainstorm narrative ideas that utilize the sample video collection. Some ideas from participants include a video starting from wide-angle shots and then showing details of single activities, a video that first shows campus life and then has a fancy transition to life during the holidays (traveling, concerts), and a video introducing various Japanese cuisines.

Familiarize with video footage. Participants were able to use Project Panel to familiarize themselves quickly with the video collection. P3 expressed that “the beginning process of editing a video can be really daunting” and Project Panel “takes a lot of the time out of needing to familiarize yourself with the content that you have”. N3 enjoyed using Project Panel as a “good preprocessing step” and expressed that it “can save people time trying to eyeball videos manually themselves.” Participants felt that Project Panel “corresponds well with [their] mental model of how [they] would look for footage (e.g., grouping common themes) (N4)” and facilitates a “faster and more intuitive way to explore a collection of video assets (P4)”.

Organizing large projects. Participants noted that Project Panel could be especially useful for organizing large projects, such as being “a unified hub for large projects involving many team members (P1)”. In particular, many participants enjoyed that Project Panel helps save time and effort required to manually assign tags to video clips or bin video clips into folders (P1, P3, P5, P6, P7, N7): “It takes forever to tag clips. It's either that or subfolders. It's such a pain to go through a whole day's worth of shooting. (P7)”

Uncover structure. Participants stated that Project Panel helps “reveal underlying structure within the recorded content (P4)”, especially those that “may not be entirely apparent at first” (P4). Participants enjoyed the “inspiration and surprise factor (N5)” and helps create “a fresh perspective on clips they might have passed up (P7)”: “Getting a new take or generating different ideas is what AI is great for and is how we can use tools like these to enhance our editing. (P7)” Participants were also able to use Project Panel to easily identify outlier videos (P2, N5): “This one color is definitely sticking out (P2)”.

Support visual ideation. Participants expressed that having the videos visually arranged in a 2D space helped them “have a better sense of the clips that they are working with and see the overall vision of what they want to create (P5)” (Principle 1). For example, some participants wanted to create a video that takes one scene from each cluster on the map (P1, P5) while others wanted to create a video that focused only on one cluster on the map (P6, N2). N1 noted that “seeing similar footage organized in the same place helps [them] figure out where to look if [they are] searching for certain elements (N1)”.

Plan the shots. Participants reported that Project Panel helped them with planning the (additional) shots that need to be taken (P5, N2). P5 stated that the “biggest pro [of Project Panel] was seeing the overall visuals of clips and the amount per grouping. I would use this at the very beginning of my workflow where I decide what other clips I need to take to complete the story. For example, if each scene of a trip is in there or if I need more concert clips.”

Prompts. Participants were able to make use of the prompt box in Project Panel for various purposes, such as quickly searching for clips, searching for similar clips, and searching for cinematic elements. As part of the first task, we asked participants to write down three prompts they found the most interesting during their explorations. The average length of the prompts is around 2 words (μ =2.24, σ =2.00). The most popular prompt was food, which was included by 5 participants. We grouped the prompts into 6 categories. Overall, participants searched for

objects 13 times (e.g., bike), scenery 10 times (e.g., outside in a grassy field), location 6 times (e.g., restaurant), action 7 times (e.g., cooking), time 3 times (e.g., daytime), and cinematography elements 3 times (e.g., establishing shot). Some example prompts from participants include videos of people (P2), running (N2), blue sky (P6), autumn colors (N5), motion blur (N5), and establishing shot (P3).

Search for a clip (instantly). Participants used the prompt box to search for clips quickly: "I would be editing at the hotel after a day of filming. Suppose I went to Rome today and I remember I went to the Trevi fountain. It'll be way easier to find it [by searching 'trevi fountain']. (P2)" P5 commented that they could use the prompt box to more easily search for "clips with triangular elements while not remembering where those clips were" and P3 commented that they could use it to "find good b-roll shots" for interview-style videos.

Search for similar clips. P7 commented that they could use the prompt box to search for similar clips: "There's a concert clip. I wonder if there are any other concert-related clips."

Search for cinematic elements. P3 and P4 commented that they could use it to find specific cinematic elements to use for their videos, such as an "establishing shot to start off a video (P3)" or a "wide-angle shot to sprinkle in some drone footage (P4)".

Swappable Lenses. Participants found the Swappable Lenses concept useful and expressed that it helped shine a light on different perspectives and allowed VideoMap to be able to cater to people with different editing styles. Participants came up with many creative ways of utilizing Swappable Lenses. P4 used the color lens to find videos of similar seasons, such as autumn. P5 used the color lens to arrange videos chronologically, starting with the day and ending with night scenes. P7 used the color lens to identify videos with warmer and cooler tones. N5 used the color lens to identify black frames to serve as transition points. N6 expressed the potential of using the color lens to maintain a steady color grading scheme. P7 used the shape lens to find videos with similar horizon positions. N6 used the shape lens to identify videos that contain

people running. P4 used the shape lens to find videos containing circular shapes. P5 used the shape lens to find videos that have similar angles and are rotating in similar motions.

Shine light on different perspectives. Participants expressed that Swappable Lenses helped shine light on new perspectives (N1, N4, N5, P7): “They give different perspectives that focus on different traits of the footage (N1)”. For example, the color lens helped reveal videos with the same color palette or aesthetic (P1, P3, P4, P6): “In the color lens, I could see the videos that have warmer colors versus cooler colors, which I didn't notice this in the semantic lens. (P7)” In addition, participants found the shape lens useful for finding videos containing people doing similar actions (N1, N5, N6, P5, P7) or videos with similar compositions (P4, P7). P7 felt that “the different lenses created subtle filters to the footage which inspired [their] decision-making process.”

Something for everyone. Participants expressed that Swappable Lenses allow VideoMap to be flexible in supporting editors who might prefer different types of editing techniques: “It also serves users’ preferences in different video editing techniques. Users can choose ways that work the best for them. (N3)” Among the 14 participants, 8 participants preferred using the semantic lens the most, 3 participants preferred using the color lens the most, and 3 participants preferred using the shape lens the most.

Paths Explorer. Participants were able to use Paths Explorer to identify suitable video transitions by recognizing interesting match cuts with minimal time and effort (RQ2). In addition, participants also expressed that Paths Explorer also helped encourage and facilitate more exploration. As part of the second task, we asked participants to record three video transitions they found the most interesting during their explorations. Some video transitions from participants include using the shape lens to cut between scenes that have buildings with similar architectural designs, using the shape lens to cut between scenes with similar composition (both having two prominent blocks of color divided at the horizon), using the semantic lens to

cut between food scenes, and using the color lens to cut between scenes featuring trees with similar autumn tones.

Identify match cuts. Participants expressed that Paths Explorer is a “cool and unique way to conceptually look at footage and generate match cut ideas (P7)”. Participants were able to easily link together different environments with similar vibes (Principle 2) (P1, P7, N1, N2, N3, N7): “[Paths Explorer] helps find associated concepts across different videos more easily. (N2)” Non-professional editors found that Paths Explorer changed how they think about video transitions. For example, N3 noted that Paths Explorer “made [them] put more thought into transitions” where they “previously had not thought too much about them.” N1 commented that Paths Explorer “inspired a lot of new ideas about how a transition could be.”

Save time and effort. Participants noted that Paths Explorer helped save time and effort in finding transitions (P3, P7, N1, N5): “Scrolling over footage and going back and forth can be very time-consuming and tedious (P7).” P3 commented that “having a lens that specifically found the paths between matching shapes shaved off hours of looking through footage.” N5 appreciated being able to see “the exact frames that can be connected together”.

Encourage exploration. Participants felt that Paths Explorer helped encourage more exploration compared to traditional video editing software (Principle 3): “It's a more exploratory approach than with the Premiere interface. I can discover relationships between scenes easier. It's a great way to anticipate multiple possible transitions without committing to an idea. (P4).” N1 appreciated the ability to view multiple transition suggestions together: “Oh these are kinda cool. This is probably better. I'll choose this one. (N1)” N2 enjoyed the ability to “simultaneously explore frames at different time points and not being constrained by the temporal ordering of video content.”

Route Planner. Participants were able to use Route Planner to quickly prototype rough cuts with automatic editing of clips (RQ3). In addition, participants also commented that Route Planner is

able to help their editing workflows by generating ideas, facilitating quick previewing of ideas, and suggesting enhancements. As part of the third task, we asked participants to record a rough cut generated by Route Planner that they found the most interesting. Some rough cuts by participants include using the shape lens to generate a video of a person running with different backgrounds, using the shape lens to generate a video with a seamless foot transition, and using the color lens to generate a video that starts from daytime and ends with nighttime.

Automatic editing. Participants enjoyed being able to quickly generate edited rough cuts using Route Planner (P2, P3, N1, N2, N3, N6): “It reduced the hassle of trying to arrange clips in a way that makes visual sense. Using traditional software would have made this process take hours. (P3)” P2 and N6 commented that Route Planner could be helpful for publishing video content on social media platforms: “It could be great for platforms that need content generated quickly, like TikTok. It could easily help create content like a supercut or quick preview. (P2)” Participants felt that Route Planner generated quality cuts: “It's a very cool video. That foot part didn't even seem like a transition. It just seemed so seamless that I thought it was the same video. (N3)”

Generate ideas. Participants commented the Route Planner can be used for idea generation. For example, N1 commented that they could throw clips into Route Planner before jumping into editing to “quickly plan out an editing plan.” On the other hand, N5 commented that they would primarily use Project Panel and Paths Explorer and “use Route Planner if [they] run out of creative ideas.”

Preview ideas quickly. Participants felt that Route Planner helps editors “try out different ideas faster than the traditional software (N6)” (Principle 3). P7 commented that “[Route Planner] was a really good way for [them] to visually come up with and execute ideas. When using Route Planner, I had thought of many different ways the videos and clips could have been edited together.”

Suggest enhancements. Participants noted that Route Planner could be used as a tool for suggesting editing enhancements. P7 commented that they could first “manually do a rough cut, then give Route Planner the clips that [they] used and see if it could come up with a better cut.”

3.2.7 Discussion

Participants generally felt that VideoMap offers a user-friendly approach to video editing (P3, P5, P6, N1, N2, N6). Specifically, non-professionals remarked that VideoMap “makes editing tasks much easier (N6)”, possesses a “smoother learning curve (N1)”, and offers a “more fun way to edit videos (N2).” Additionally, participants expressed that VideoMap helps reduce the non-creative grunt work involved in video editing, allowing for greater focus on creative aspects (P5, N5, N6): “Trying to write my process in Premiere made me realize how complicated it actually is to do this task, [and] how much manual searching and scrolling through frames I do. (P5)”. N6 explained that their workflow of organizing video footage using traditional editors involves “manually sorting the videos in a file browser, picking the videos that [they are] most interested in, then manually uploading them into the editing software (N6)”. With Project Panel, they could “have all videos uploaded at once, then filter with prompts using the ideas [they] have in mind (N6)”.

One clear benefit of VideoMap is its enhanced overview capability (Principle 1). By breaking the sequential nature of browsing videos, viewing videos across a latent space in the Project Panel helps editors explore raw footage, enabling them to better familiarize themselves with and uncover structure within the footage (RQ1): “[VideoMap] helped me get a more comprehensive overview of the materials more quickly (P4)”. P7 reflected: “I think the traditional video editor can leave much to be desired. [VideoMap] is a fresh take on the old concept, allowing for the clips to shine and visually show [me] what they are and how [I] might think to compose them.” N5 commented that “[VideoMap] is a more inspirational way of organizing video clips” and N3 noted that “[VideoMap] helps editors tell better stories with connected themes, color, and shape”. Given an enhanced overview, participants were also able to visually ideate narratives,

such as creating a video that weaves together one clip from each cluster on the map (P1, P5), or conversely, creating a video that only focuses on clips from one cluster (P6, N2). In addition, P5 was also able to utilize the overview to “decide what other clips [they] need to take to complete the story”, such as capturing more concert footage after noticing relatively small semantic concert clusters on the map.

By operating on the latent space, VideoMap helps visualize semantic or visual neighbors, thereby incentivizing editors to maintain continuity in their edits (Principle 2). Paths Explorer assists editors in brainstorming opportunities to create continuous transitions through transition recommendations to latent space neighbors (RQ2): “[Paths Explorer] is a cool and unique way to conceptually look at footage and generate match cut ideas (P7)”. N5 explained that “using a traditional video editing software might involve jumping back and forth between the videos or overlaying frames on top of each other with a lower transparency to see if there are some common shapes”. P3 contrasted editing transitions using Paths Explorer with their typical workflow: “It really changes the process of editing. Usually, when you're doing this [with traditional editing software], you already have everything onto a timeline, and you're like, alright, now let's find the transitions.” Furthermore, non-professionals felt that Paths Explorer “made [them] put more thought into transitions [where they] previously had not thought too much about them (N3)”, and helped “inspire a lot of new ideas about how a transition could be (N1)”.

VideoMap enables a more exploratory approach to editing through various components and mechanisms, such as automatic rough cuts with Route Planner, map-inspired navigation elements, prompt searching in Project Panel, and recommending multiple possible transitions in Paths Explorer (Principle 3). By automatically generating rough cuts, Route Planner helps editors quickly test and preview editing ideas (RQ3). Participants utilized the generated rough cuts for idea generation (“quickly plan out an editing plan (N1)”), previewing the results of editing ideas (“I had thought of many different ways the videos and clips could have been edited together (P7)”), and suggesting enhancements (“see if it could come up with a better cut (P7)”).

Participants commented positively on various interactions and map-inspired navigation elements, such as scrubbing landmarks to play through videos (P5, P6, N6) and selecting nodes to visualize paths to neighbors (P4, N1, N2). Participants enjoyed using prompts in the Project Panel to semantically search for clips in mind or discover new clips: “It is like the difference between using Google Search and searching information from books (N2).” Additionally, P4 commented that “Paths Explorer helped [them] discover and plan possible transitions much faster” and felt like a “more exploratory approach than with [a traditional editing interface]”, given the ability “to anticipate multiple possible transitions without committing to an idea”.

Finally, we envision a vast and versatile design space for VideoMap, with future developers enhancing and extending its capabilities. Our current proof-of-concept enables editors to edit video through notions of semantics, color, and shape. We plan to open-source our code so that developers can build new lenses to support additional editing tasks. Furthermore, we hope that VideoMap can be used as a building block to support the development of new creative video editing applications. In the following section, we demonstrate some examples.

3.2.8 Applications

We implemented three extended applications of VideoMap to demonstrate how it can be customized for additional use cases. These applications include (1) video summarization with semantic districts, (2) video highlights with custom landmarks, and (3) text-based video editing.

Video Summarization

We show how VideoMap Project Panel can be extended to help editors create summary videos (Figure 7). We first project the source video's frames into a semantic latent space. Instead of creating districts per video, we apply k-means clustering to automatically partition the video frames into “semantic districts.” Since we are working in a semantic space, each district approximately represents a main activity of the source video. We determine the number of districts (i.e., k value) using the elbow method [68]. We color-code the districts and visualize the centroid frame of each district as semantic landmarks. To create the summary video, the editor

can add semantic districts (i.e., main activities) by clicking on the landmarks, in the order they want them to appear. Selected landmarks are highlighted with red borders (Figure 7 a) and displayed as a storyboard at the bottom (Figure 7 b). We generate the summary video by taking three seconds of video centered around the landmark for each semantic district).

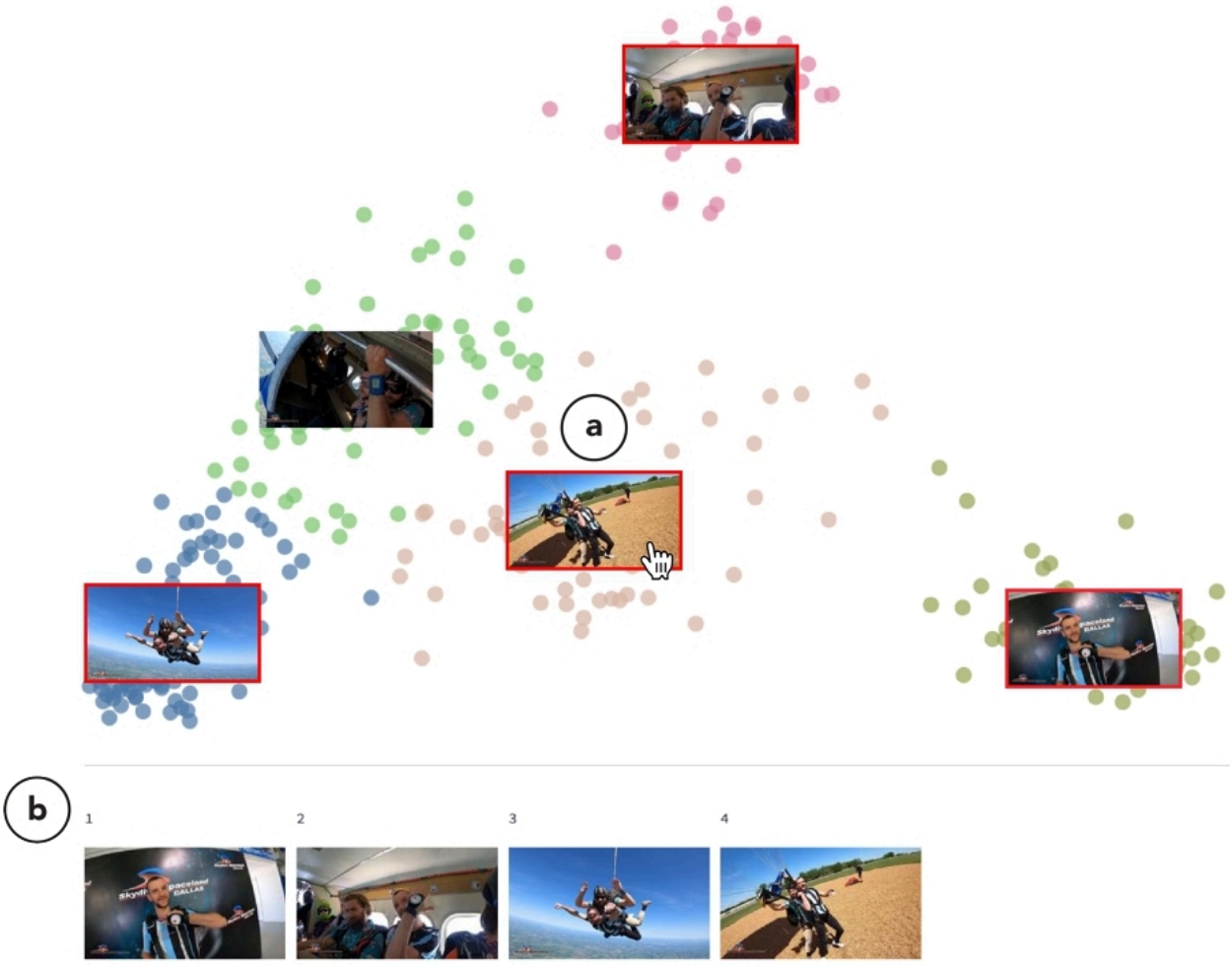

Figure X: VideoMap's Project Panel can be extended to create summary videos. We automatically create “semantic districts” that approximately represent the main activities of a video using k-means clustering under the semantic lens. The editor can select several landmarks to specify the activities to include in the summary video. Selections are highlighted with red borders (a) and displayed as a storyboard (b).

Video Highlights

We show how VideoMap Paths Explorer can be extended to help editors create highlight videos (Figure 8). The editor can first upload a photograph depicting an activity (e.g., skydiving). We

then project the photograph into the latent space under the semantic lens and visualize it as a “custom landmark” (Figure 8 a). Next, we draw a path from the custom landmark to the nearest video frame in the latent space (Figure 8 b). Videogenic [46] found that photographs taken by photographers tend to capture the most highlight-worthy moments of an activity (e.g., when the skydiver jumps out of the aircraft). Thus, we take the photograph's neighboring video frames as the highlight video. We generate the highlight video by taking five seconds of video centered around the nearest video frame).

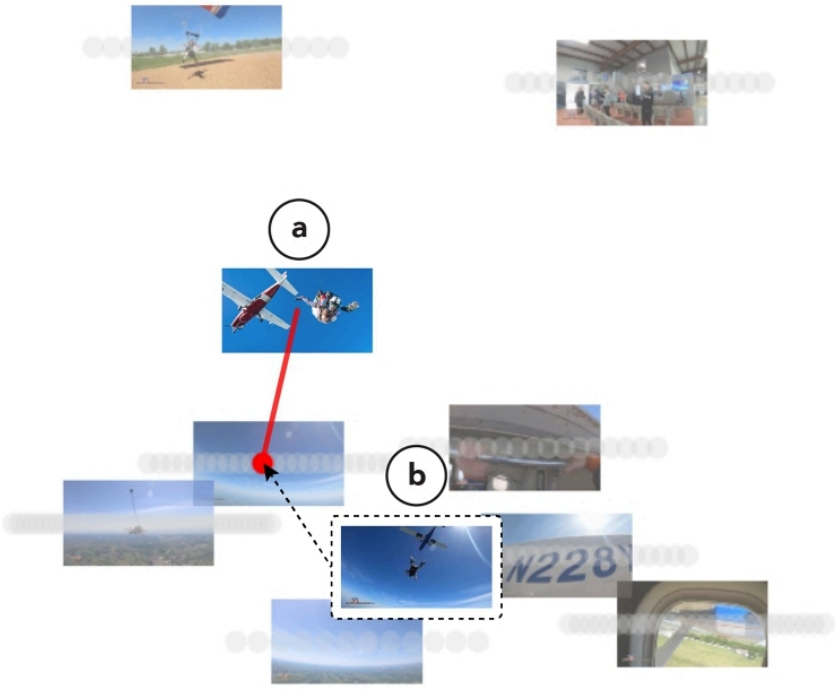

Figure X: VideoMap's Paths Explorer can be extended to create highlight videos. The editor can upload a photograph (i.e., a custom landmark) depicting an activity (e.g., skydiving) (a). Our key insight is that photographs taken by photographers tend to capture the most highlight-worthy moments of an activity (e.g., when the skydiver jumps out of the aircraft). We then generate a highlight video using near neighbor video frames to the custom landmark in the semantic space (b).

Text-Based Video Editing

We show how VideoMap Route Planner can be extended to support editors in text-based video editing (Figure 9). The editor can first describe the video they want to create using descriptive sentences, like writing a story (Figure 9 a). For each sentence, we match the closest video in the

semantic space, using the same CLIP-based method for comparing text and image embeddings as detailed in our implementation for Prompts (see Section 4.1.1). We then create the shortest route along the videos using the same dynamic programming approach as detailed in Route Planner (see Section 4.4) with an additional ordering constraint (i.e., in the order of the story) (Figure 9 b).

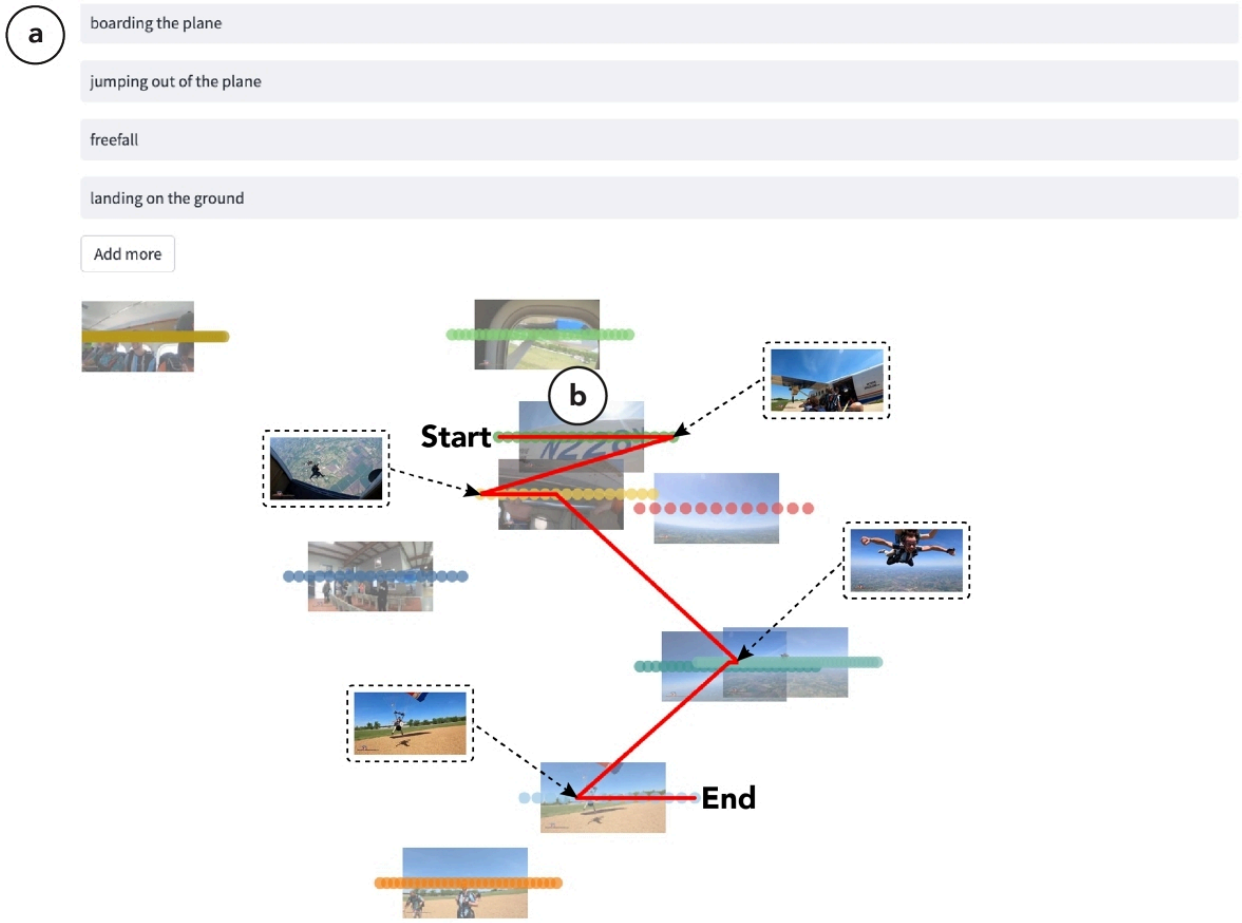

Figure X: VideoMap's Route Planner can be extended to edit videos using text. The editor can describe a desired video using descriptive sentences, like writing a story (a). We then match each sentence to the closest video clip in the semantic space and generate a video by finding the shortest route along the clips (b).

3.2.9 Limitations and Future Work

While VideoMap was positively received in our user study, there are several avenues for improvement that we plan to address for future work. First, VideoMap currently supports three

types of lenses that facilitate the formation of narratives (semantic) and the creation of visual transitions (shape and color). In our study, participants suggested additional lenses, such as a cinematography lens to cluster videos by different shot types (e.g., by training a shot type classification model), an Image Signal Processor (ISP) lens to cluster videos by white balance, focus, and exposure values (e.g., by utilizing image analysis algorithms), and a motion lens to cluster videos by distinct camera movements (e.g., by computing motion vectors with optical flow [50]). For future work, we plan to open-source the pipeline code for developers to build new lenses by generating new latent spaces that are meaningful for video editing. This pipeline could involve generating embeddings using AI foundation models [11] or training custom models tailored to specific video editing objectives. Moreover, we could also explore the creation of "hybrid lenses." For instance, we could create a lens with weightings of 80% shape and 20% semantics to enable the discovery of shape-matching cuts that also possess related semantics. Hybrid lenses could provide editors with the flexibility of blending together the properties of multiple lenses, allowing them to fine-tune the latent space according to their editing goals. Second, due to time constraints, the current design of our user study consists of users trying out VideoMap without a comparative baseline system. For future work, it may be interesting to compare editing using VideoMap against editing on traditional sequential editing timelines and analyze how VideoMap could change the way people engage in video editing. Finally, while VideoMap currently caters to primarily maintaining the continuity principle in video editing, sometimes editors may wish to intentionally break this principle and connect two contrasting scenes to create a shocking effect. To support this, we may extend the Paths Explorer to find far-away cuts. Being able to find contrasting yet sensible transitions could also be an interesting exploration area.

3.2.10 Summary

VideoMap generates a navigatable *map* of video collections by projecting video frames in *latent space*, transforming video editing from sequential timeline seeking to spatial exploration with swappable lenses. VideoMap primarily improves the *inspectable* characteristic of AI controllability (constrained view → overview) and shifts upwards in *intuitiveness* in the

intuitiveness-controllability spectrum. VideoMap demonstrates the power of *swapping* between different task-specific AI models. In the next chapter, I explore helping designers *combine* multiple task-specific AI models to create complex multimedia workflows.

3.3 Jigsaw: Puzzle Pieces as an Interface

This chapter was adapted from my published paper: Lin, D. C. E., & Martelaro, N. (2024, May). Jigsaw: Supporting designers to prototype multimodal applications by chaining AI foundation models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (pp. 1-15).

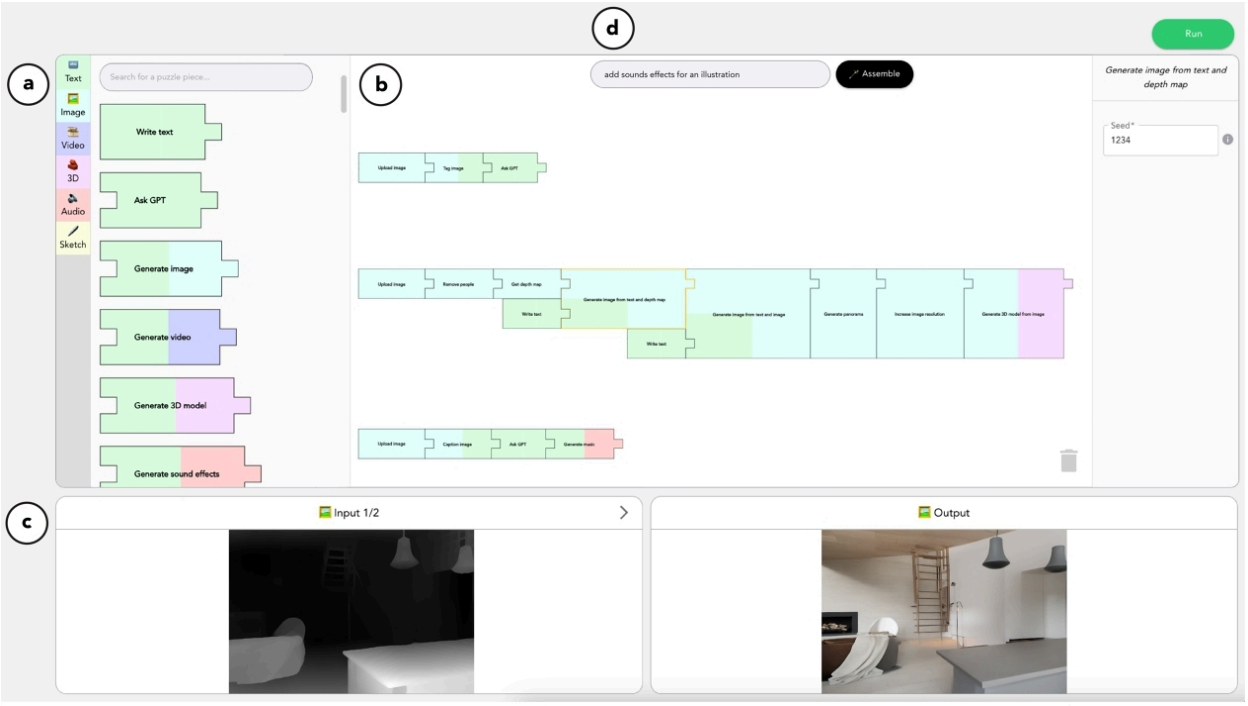

Figure 1: Jigsaw is a prototyping system that lets designers generate creative content with AI foundation models represented as puzzle pieces. Designers can first search for model capabilities via the Catalog Panel (a). Designers can drag corresponding puzzle pieces onto the Assembly Panel (b) and combine different AI capabilities across varying modalities into a chain by assembling compatible pieces. Designers can specify inputs and observe the intermediate results of a chain via the Input and Output Panels (c). Designers can ask the Assembly Assistant (d) to recommend a chain of AI models to accomplish a task.

3.3.1 Introduction

The past year (2023 as of the writing of this paper) has seen substantial progress in the capabilities of AI foundation models [13]. These models, which are pre-trained on vast quantities of data, can perform many tasks “off the shelf” without further training. Consequently, many foundation models essentially become input-output systems, simplifying the complexities of working with AI by abstracting models to their core capability [53]. Until

recently, creating an AI-enabled system necessitated users to curate their own data, train a model, and occasionally modify the model architecture to adapt to their use cases [10].

With powerful plug-and-play capabilities, many designers have begun embracing AI foundation models to enhance their creative workflows. New foundation models support a wide variety of tasks and modalities, including large language models [43] such as GPT [14] for text generation and processing, image generation models such as Stable Diffusion [36], image segmentation models such as Segment Anything [26], and models for video [42], 3D model [24], audio [28] generation.

However, despite the variety of capabilities offered, the integration of foundation models within the creative process can be challenging. Our initial observations suggest that these models are often used for one-off tasks or as standalone applications. For instance, a designer might use ChatGPT [5] to brainstorm and generate ideas, or they might use Midjourney [7] to generate visual prototypes. To incorporate the results of these models into their broader creative process, designers manually copy and paste the results into another design tool. Moreover, despite the variety of available models, designers typically only use a small selection of highly publicized models (ChatGPT, Stable Diffusion, MidJourney) and are often unaware of the range of capabilities and modalities they could potentially utilize from lesser-known models.

To gain a deeper understanding of designers' challenges when using current AI models in their creative processes, we conducted a formative study with ten designers. From our formative study, we identified four key challenges:

1. Designers are often unaware of the full range of capabilities offered by different types of foundation models.
2. Designers struggle with the need to be "AI-friendly," which includes difficulties in forming effective prompts and selecting optimal parameters.
3. Designers find it challenging to cross-integrate foundation models that exist on different platforms and are specialized for different modalities.

4. Designers find prototyping with these models to be a slow and arduous process.

Based on the findings from the formative study, we derived four design goals, which informed the development of Jigsaw, a block-based prototype system that represents foundation models as puzzle pieces and allows designers to combine the capabilities of different foundation models by assembling compatible puzzle pieces together. Jigsaw includes features that help designers discover available foundation model capabilities and find the right model for their use case. Jigsaw also includes "glue" puzzle pieces that translate design ideas into prompts for other models, clear explanations of parameters to help users make model adjustments, and an Assembly Assistant that recommends potential combinations of foundation models to accomplish a task specified by the designer. To assess the utility of Jigsaw, we invited ten designers from the formative study to test the system. We evaluate how well designers create creative AI workflows given a design brief and during free exploration. The results show that Jigsaw helps designers better understand the capabilities offered by current foundation models, provides intuitive mechanisms for using and combining models across diverse modalities, and serves as a visual canvas for design exploration, prototyping, and documentation.

This research thus contributes:

- A formative study with ten designers that identifies the challenges designers face when using AI foundation models to support their work.
- Jigsaw, a prototype system that assists designers in combining the capabilities of AI foundation models across different tasks and modalities through assembling compatible puzzle pieces.
- A user study that demonstrates the utility of Jigsaw to designers and informs areas for future block-based prototyping systems for prototyping with AI foundation models.

3.3.2 Related Work

This work draws on prior research in AI foundation models, visual programming interfaces, and designer-AI interaction.

AI Foundation Models

The term "foundation models" characterizes an emerging family of machine learning models [13], often underpinned by the Transformer architecture [41] and trained on vast amounts of data. The researchers who introduced this term defined foundation models as "models trained on broad data (generally using self-supervision at scale) that can be adapted to a wide range of downstream tasks." [8] The strength of foundation models lies in their capacity for out-of-the-box usage across various tasks. This signifies an improvement from the previous AI landscape, where users had to create their own datasets for custom use cases and fine-tune models [10].

Prominent examples of foundation models include large language models such as GPT [14] which can perform a variety of text generation tasks and image generation models such as Stable Diffusion [36] which can generate a diverse range of images from text-based prompts. Foundation models also go beyond generative models and include models for tasks such as classification [33], detection [57], segmentation [26], spanning a range of modalities including text [14], image [36], video [42], 3D models [24], and audio [28]. Many foundation models perform tasks across modalities, such as text-to-x generative models and x-to-text classification models. In turn, this allows foundation models to be treated as x-to-x input-output systems. Such abstraction greatly simplifies how people can use and combine such models in larger AI-enabled systems. Our research aims 1) to inform designers about the capabilities offered by foundation models that can be useful for creative tasks, and 2) to incorporate these capabilities into their creative workflows. In particular, we are interested in exploring how designers can combine the capabilities of multiple models across different tasks and modalities by connecting them together on a visual interface.

Visual Programming Interfaces

Visual programming interfaces (VPIs) have been extensively studied as tools to aid users in designing and implementing systems through graphical elements rather than text-based code [31]. A key benefit of VPIs is their lower entry barrier for novice programmers [45]. There are primarily two main paradigms for VPIs. The first, the dataflow paradigm, lets users specify how a program transforms data from step to step by connecting nodes in a directed graph. Pioneering work in this area includes Prograph [17] and LabVIEW [27]. The second paradigm utilizes block-based function representations and lets users create programs by connecting compatible components together. Notable works in this area include Scratch [35] and Blockly [19]. Many commercial creative applications have adopted VPIs, including game engines such as Unity [11], CAD tools such as Grasshopper [9], and multimedia development tools such as Max/MSP [12].

VPI concepts have been applied to machine learning applications. For example, Teachable Machine [15] uses a visual interface to help students learn to train a machine learning model. ML Blocks [46] assists developers in training, evaluating, and exporting machine learning model architectures. Very recently, researchers in both academia and industry have worked on VPIs that support the creation of AI workflows through the combination of pre-trained models. Several works have investigated node-based interfaces for building Large Language Model (LLM) pipelines, including PromptChainer [48], FlowiseAI [2], and Langflow [3]. Most closely related to our work are Rapsai by Du et al. [18] and ComfyUI [1]. Both tools provide a node-based interface for machine learning researchers and enthusiasts to build multimedia machine learning pipelines. These tools are catered more toward users with at least some background knowledge in AI programming, giving users the flexibility to customize the tools through programming at the expense of exposing more technical elements to the user.

Our work builds upon prior and concurrent VPI tools and research. However, we made several design choices for our tool to help better support non-technical designers (Table 1). First, our tool leverages a block-based VPI paradigm, which has been shown to be effective in supporting

novice programming learners [35]. Second, in the same spirit as other creative AI tools such as RunwayML [4], our tool supports AI capabilities of a diverse range of modalities. Third, our tool offers integrated AI assistance features for designers, such as the Assembly Assistant (Section 4.4), semantic search (Section 4.1.3), and glue pieces (Section 4.1.4). We build on recent advances in the reasoning capabilities of LLMs to power these features [47]. To the best of our knowledge, this research is the first to study 1) supporting non-technical designers in prototyping design workflows with AI through a block-based visual interface and 2) utilizing the plug-and-play capabilities of AI foundation models that have emerged over the past year, covering a diverse range of tasks and modalities.

Table 1: Comparison of Jigsaw against related tools. Jigsaw supports non-technical designers with a beginner-friendly block editor and offers AI capabilities across multiple modalities. Jigsaw's Assembly Assistant can help automatically recommend a chain of AI models for a designer-specified task.

Tool name	Interface design	Supported modalities	Target audience	Assembly assistant	Release year
Jigsaw	Block editor	Text, Image, Video, 3D, Audio, Sketch	Designers	Yes	2023
PromptChainer	Node editor	Text	Developers	No	2022
FlowiseAI	Node editor	Text	Developers	No	2023
Langflow	Node editor	Text	Developers	No	2023
Rapsai	Node editor	Text, Image	ML researchers	No	2023
ComfyUI	Node editor	Text, Image, Video	ML enthusiasts	No	2023
RunwayML	Standalone models	Text, Image, Video, 3D, Audio	Designers	No	2018
Visual ChatGPT	Chatbot	Text, Image	General public	Yes	2023

Designer-AI Interaction

Several works from the HCI design community have examined the ways in which designers perceive and interact with AI. Chiou et al. [16] follow a Research through Design (RtD) [58] approach and find that AI can offer designers new perspectives and avenues of design

exploration. Shi et al. [39] conduct a landscape analysis of AI and suggest the opportunity to build more tools that enable co-creativity between designers and AI. Yang [49] proposes the vision of designers working with AI as a "design material". This research follows this thread of work to build a tool to help designers prototype new design workflows using AI and with the support of AI.

Subramonyam et al. [40] argue that a challenge with using AI as a design material is that the properties of AI only emerge as part of user experience design. They thus employ data probes with user data to help elicit AI properties and facilitate working with AI as a design material. Yang et al. [51] identify that designers often find designing with AI difficult due to uncertainty about the AI's capabilities and the complexity of the AI's outputs. Gmeiner et al. [20] identify the primary challenges for designers when co-creating with AI design tools as understanding and manipulating AI outputs and communicating design goals to the AI. In this research, we offer mechanisms to help designers overcome these challenges, such as conveying AI capabilities, supporting easy inspection and manipulation of AI outputs with real data, and allowing users to communicate design goals to the AI using natural language.

Liu et al. [30] find that when designers use the "right" prompts, they achieve significantly higher quality results from generative models. However, Zamfirescu et al. [54] find that people generally struggle with writing effective prompts. In this research, we introduce a puzzle piece (translation glue) to help designers automatically translate pieces of text into prompts. Yang et al. [50] find that designers are more successful when they collaborate with data scientists. Using RtD, Yildirim et al. [52] identify that designers develop boundary objects to communicate design intentions with data scientists. In this research, we let designers document their creative process on a canvas (Assembly panel), which designer participants in our user study found to be a useful boundary object for sharing and explaining ideas.

3.3.3 Formative Study

We conducted a formative interview study with ten designers to understand how designers attempt to use AI in their work and inform the development of a new tool to support creative work with AI.

Participants and Procedure

We interviewed ten designers (P1-P10, 6 male and 4 female, aged 24-39), recruited through known contacts and word of mouth. The designers come from diverse specializations, such as interior design, product design, graphic design, and video game design. All participants have more than five years of design work experience and use AI tools to support aspects of their design processes, such as ChatGPT, Midjourney, and DALL-E [34]. We conducted one-hour interviews remotely over video conferencing, asking participants to describe their typical creative workflow, how they use AI to support their work, the specific AI tools they use, and the pain points they face using AI. Following the interviews, the first author conducted a thematic analysis of interviews and summarized participants' key challenges.

Findings and Discussion

We identify four key challenges designers face when using AI to support their work.

C1: Limited Knowledge of AI Capabilities. Despite the broad spectrum of AI foundation models available, designers felt they had limited knowledge of existing models and their capabilities. As a result, they felt like they were underutilizing the creative support these models could provide. Designers found it challenging to "understand the capabilities of various models in a crowded market (P1)", making it difficult to determine which model is most suitable for a specific task. Additionally, designers expressed a desire to "easily view a few example results from the models (P5)", which would allow them to quickly assess the model's capabilities and determine if its results align with their intended use case.

C2: Tedious to be AI-friendly. After deciding on an AI model to use, designers stated that it is challenging to be "AI-friendly." This includes crafting effective prompts (for generative models) and setting optimal parameter values of AI models to ensure good results. Designers stated it can be time-consuming to "master the art of prompt creation (P2)", often dedicating a significant amount of time to simply translating their design idea into a functional prompt. As P5 stated, "behind every stunning image generated by Stable Diffusion lies a designer's patience and a relentless pursuit of the right prompt." Moreover, our participants were often confused about different model parameters and how they affect the model's results, leading to "endless parameter tweaks (P10)".

C3: Difficult to Combine Multiple Models. Designers felt that current models predominantly cater to simple and singular functionalities. Designers commented that for realistic design workflows, which involve multiple tasks and a range of modalities, they often find themselves having to switch between distinct AI platforms. This fragmented the design process, and as P9 stated, "switching between AI platforms felt like needing a different kitchen gadget for every step in a recipe." In addition, designers often face compatibility issues between models when attempting to combine them, leading to time-consuming troubleshooting. They commented that it would be beneficial to "clearly know which models are compatible with one another (P6)."

C4: Slow Prototyping and Iteration. Designers noted that "seamless prototyping and iteration is crucial to the design process (P6)". However, when working with AI, designers frequently found it challenging to quickly build prototypes and view results. Setting up and switching models can be a lengthy process that inhibits rapid experimentation. Furthermore, when creating workflows that involve chaining models, designers often can only view the final result. This makes it difficult to understand how individual models affect the final result and can make it challenging to explain design decisions to clients without tangible intermediate outputs.

Design Goals

To tackle the challenges designers encounter when using AI, we distill four design goals.

D1: Catalog of AI Foundation Models. To help designers gain a better understanding of available AI foundation models, we aim to compile a catalog of existing models. For each model, we should provide straightforward explanations of its capabilities along with examples of their results. Furthermore, we should provide mechanisms for designers to easily find models that can accomplish the specific tasks they have in mind.

D2: User-friendly instead of AI-friendly. We should provide mechanisms that reduce the need for designers to adapt to the nuances of AI models. First, we should incorporate assisted prompting techniques to help designers translate design ideas into prompts. Second, we should explain model parameters in laypeople's terms, including how altering different values will impact model results.

D3: Intuitive Interface for Combining Models. We should provide designers with an interface that allows them to easily combine multiple task-specific foundation models across a wide range of modalities. The interface should visually present clear affordances of which models can be combined. In addition, we should provide an assistive tool for suggesting model combinations.

D4: Facilitate Effective Prototyping. Designers place significant importance on experimentation and iteration. We should make it effortless for designers to experiment with different model combinations and be able to easily view results. Furthermore, we should let designers view intermediate results within a chain of models to help diagnose errors and aid in communicating design ideas with clients.

3.3.4 System Implementation

The following outlines Jigsaw's four major components: the (1) Catalog Panel, (2) Assembly Panel, (3) Input and Output Panels, and (4) Assembly Assistant. We then describe how a designer can use Jigsaw with an example interior design workflow.

Catalog Panel

The Catalog Panel assists designers in selecting suitable models for their tasks with a catalog of foundation model components (D1).

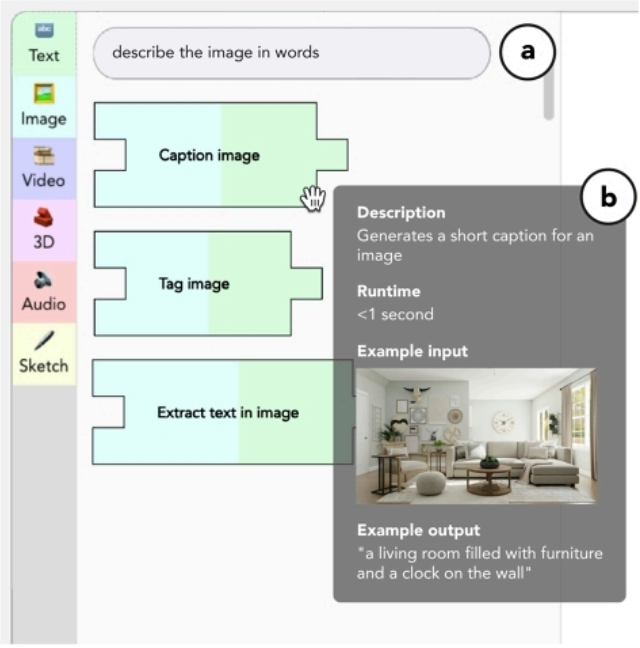

Figure X: Users can search for model pieces by describing their task in the semantic search bar (a). Users can hover over a model piece to view a description of its capability, typical runtime, and an example input and output (b).

Curating a catalog of foundation models. We identified six common modalities used in creative work, namely, text, image, video, 3D, audio, and sketches. Jigsaw curates available models across all possible pairwise permutations of modalities (e.g., text-to-text, text-to-image, text-to-video,...). Jigsaw also includes foundation models that with dual input channels, such as ControlNet [56]. For tasks supported by multiple models, we prioritize models based on: 1) inference speed (ideally less than a minute to run), 2) zero-shot capability (plug-and-play use),

and 3) the quality of results (models ranked highly on machine learning benchmarks). Overall, we implemented a catalog of thirty-nine models across six modalities (see Appendix A for a full listing).

Representing foundation models as puzzle pieces. Considering foundational models as input/output systems, we represent them as puzzle pieces with input and output arms. There are two types of puzzle pieces: 1) the model piece represents models with customizable parameters, and 2) the input piece accepts input from the user via text, media file, or sketch (see Section 4.3). Jigsaw color-codes puzzle pieces based on their input and output modalities to signals what pieces are compatible with one another (D3). For example, a text-to-image piece would be colored green on the left and blue on the right. When a user hovers over a puzzle piece, a tooltip provides a description of its capability, typical runtime, and an example of an input and output (D1).

Helping users find the right piece. Jigsaw provides two mechanisms for users to find model pieces for their tasks (D1): 1) puzzle pieces are grouped by input modality and 2) users can describe the task in the semantic search bar. The search returns model pieces with high semantic similarity to the query, scored using CLIP [33] text embeddings.

LLMs as glue. There can be instances where models do not perfectly align, such as situations where intermediate reasoning is required. Drawing inspiration from Socratic Models by Zeng et al. [55], we utilize Large Language Models (LLMs) as connecting elements between model pieces. We refer to these instances as glue pieces. The user first attaches a model piece capable of conveying the content of a modality in text (x-to-text). Next, the user attaches the LLM glue piece for language-based reasoning (text-to-text). Finally, the user attaches a model piece which translates text back into another modality (text-to-x). To help users connect model pieces in common use cases, Jigsaw includes three types of glue pieces:

- The custom glue piece accepts any custom user instruction.

- The translation glue piece converts a piece of text into a prompt that better aligns with text-to-x models (e.g., Stable Diffusion) (D2) (Figure 3 a). We ask GPT to transform an input into a prompt via the following prompt:

```
Here are example prompts for a text-to-<modality> generation
 model: <list of example prompts>.
Transform <input data> into a prompt. Answer in only the
transformed prompt.
```

- The ideation glue piece accepts a design task specified by the user and generates an idea (Figure 3 b). We ask GPT to generate an idea via the following prompt:

```
Generate an idea for <task> based on <input data>. Answer in
 one short sentence.
```

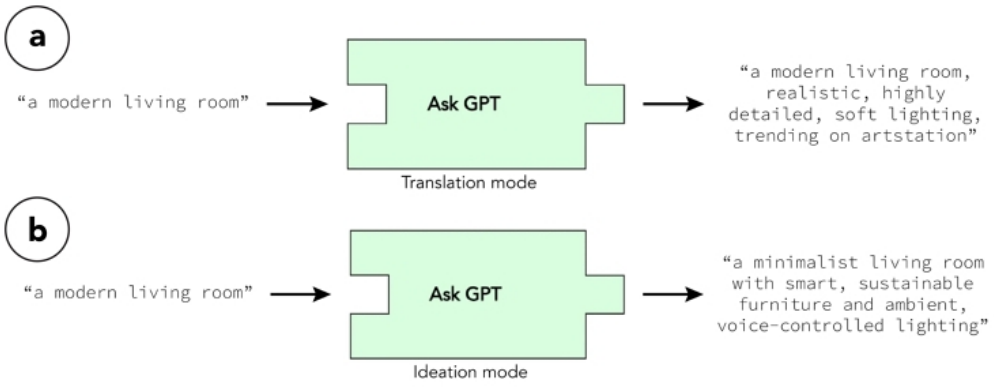

Figure X: The translation glue piece converts a piece of text into a prompt format suitable for text-to-x generation models (a). The ideation glue piece generates an idea for a design task (b).

Assembly Panel

The Assembly Panel offers an infinite canvas for combining compatible foundation model puzzle pieces (D3) (Figure 4). When the user clicks on a model piece, a parameters sidebar allows users to customize a model's specific parameters. Jigsaw pre-populates each model's parameters with default values that generally yield good results and defines limits so that the user can experiment with different values without the concern of breaking the model (D2). Tooltips explain, in plain English, how the parameter influences model results and recommends optimal

values for common scenarios. Users can build multiple chains on the canvas and can run each chain separately, allowing parallel explorations and complex workflows.

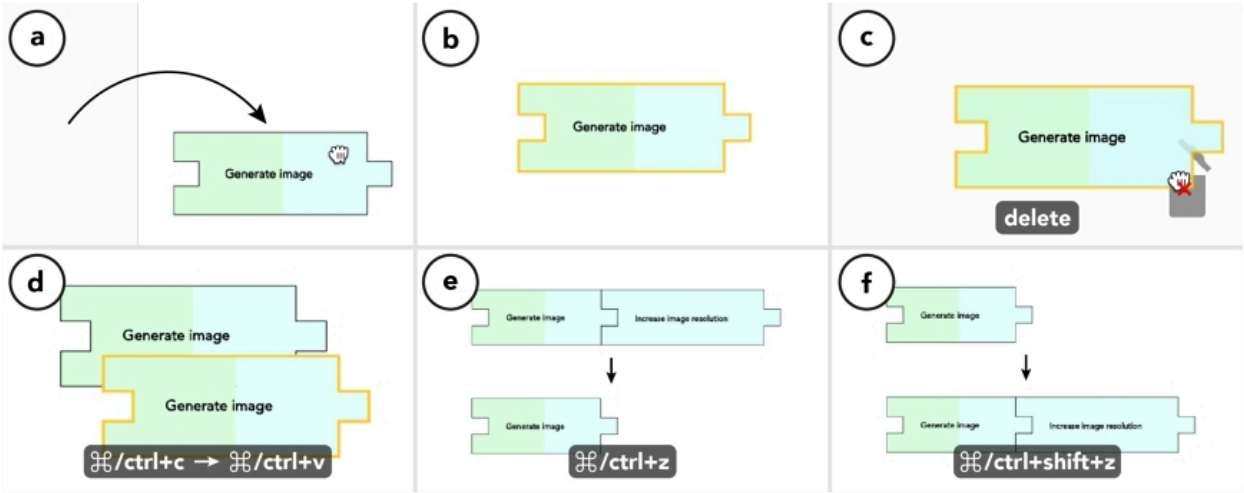

Figure X: Users can drag puzzle pieces from the Catalog Panel onto the Assembly Panel (a), select pieces on the Assembly Panel by clicking on them (b), and remove pieces by dragging them to the trash bin or pressing the delete key (c). Users can duplicate pieces, and undo and redo actions using hotkeys (d-f).

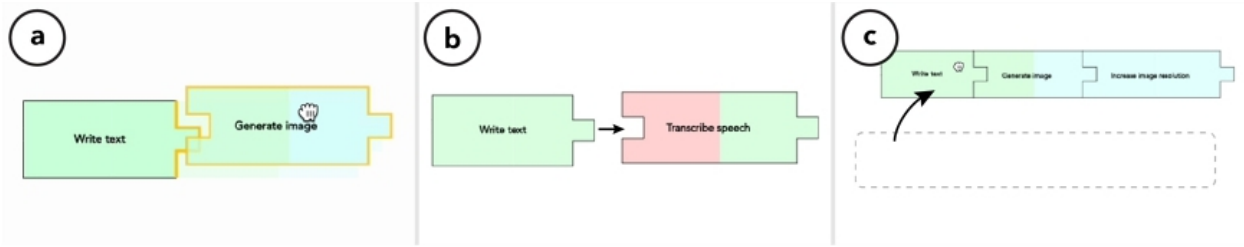

Figure X: When the user drags a puzzle piece close to another compatible piece, Jigsaw displays a semi-transparent preview of the potential connection. If the user releases the puzzle piece, it will snap into place (a). Conversely, if the user attempts to connect a puzzle piece to an incompatible piece, the new piece will be repelled, ensuring that users do not force a fit (b). Users can move multiple puzzle pieces simultaneously (c).

Input and Output Panels

The Input and Output Panels allow users to input, view, and download media across modalities. Users can type into the input panel (Figure 6 a), upload files (Figure 6 b), or draw sketches (Figure 6 c). The Output Panel shows the result of the chain and lets users copy (Figure 6 d) or download outputs (Figure 6 e-i).

The user can select a puzzle piece to view the intermediate inputs and outputs at that specific piece. This allows the user to observe how the data is transformed at each stage (D4). Additionally, within a chain, the user can view how the inputs for a puzzle piece affect the results of a puzzle piece located several steps downstream by holding the shift key to select multiple puzzle pieces.

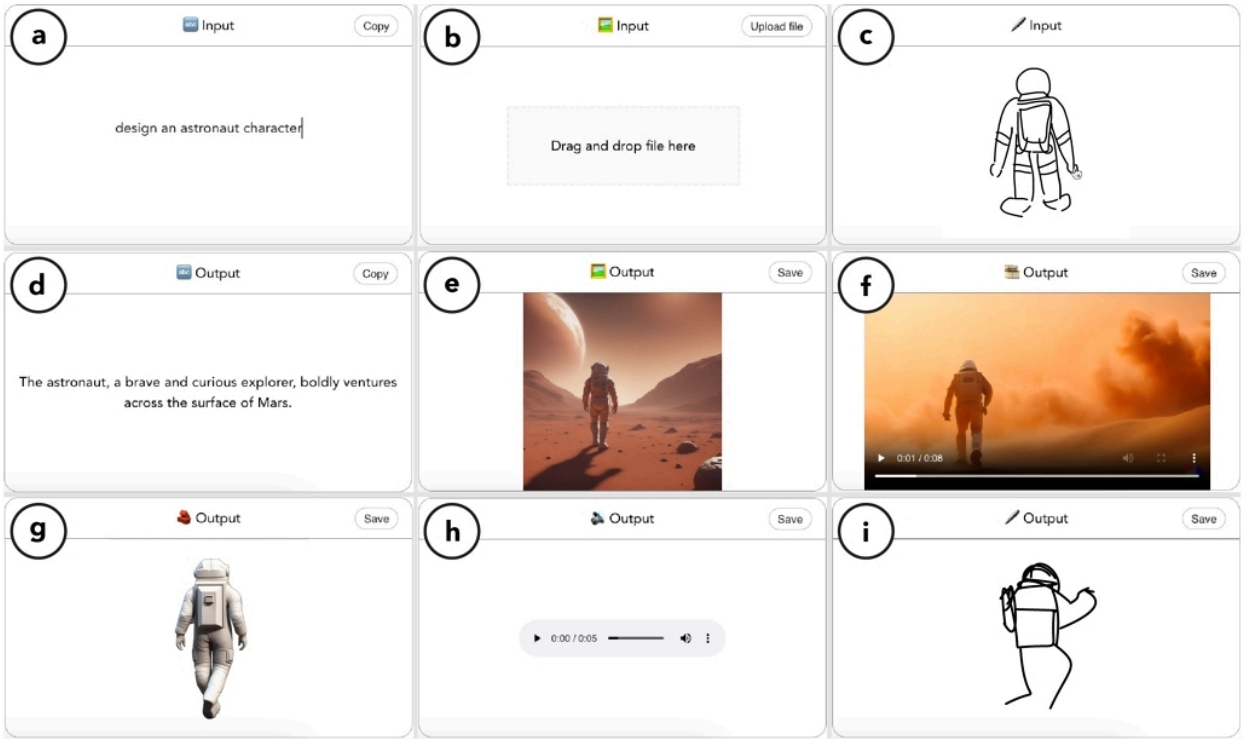

Figure X: Text inputs can be directly typed into the Input Panel (a). Image, video, 3D, and audio inputs can be uploaded either by drag-and-drop or the file browser (b). Sketch inputs can be drawn (c). Text outputs can be viewed and copied by the user (d). Image, video, 3D, audio, and sketch outputs can be viewed in their respective media viewers and downloaded by the user (e-i).

Assembly Assistant

The Assembly Assistant recommends a chain of puzzle pieces for a user-specified task. The designer would first provide a natural language description of a task, such as “Add sound effects for an illustration.” Jigsaw then asks GPT to use a chain of models to accomplish the task via the following prompt:

```
You are given a set of AI models to complete a user's task.
There are thirty-nine models: <1. text2text() has reasoning
capability. 2. text2img() can generate an image from text.
3. text2video() can generate a video from text, ...>
You can only use the models given. You do not have to use
all the models. You will answer in a JSON format. Here is an
 example answer: <example combination of puzzle pieces
written in a JSON format for the frontend to parse>. Your
task is to <user-specified task>.
```

Prior work has found that asking GPT to evaluate its own results can improve the correctness of its responses [32]. We thus ask GPT to evaluate its own answers based on four criteria via the following prompt:

```
<Prompt from the previous step>
<Answer from the previous step>
Here are four criteria that the answer needs to satisfy. If
any criteria are not satisfied, please give me the corrected
 answer in JSON format.
1. Whether the user's task was understood and completed.
2. Whether no models outside of the provided ones were used.
3. Whether the output and input of each step can be
connected.
4. Whether it follows the correct JSON format.
```

Jigsaw then passes the chain of puzzle pieces provided in JSON format to the frontend and adds them onto the Assembly Panel. The designer can make further edits to the chain, just like any manually-created chain.

System walkthrough

We illustrate the interactions supported by Jigsaw using an interior design example. Figure 7 displays the final arrangement of puzzle pieces, referred to as the "model mosaic" in this paper for succinctness.

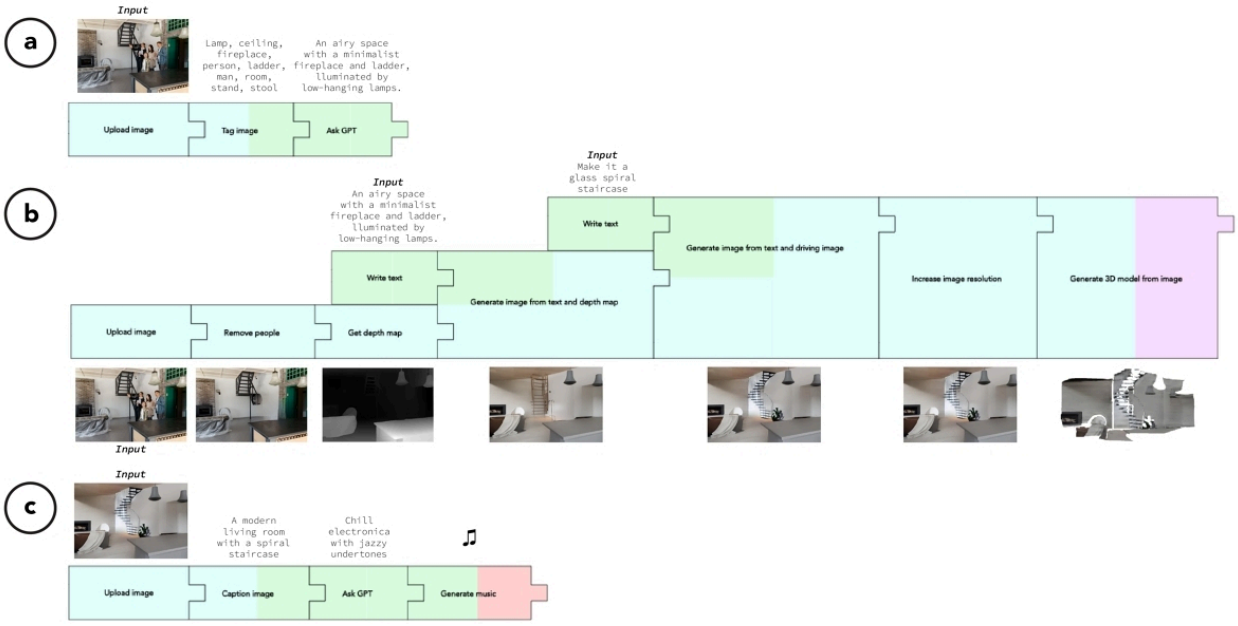

Figure X: An example model mosaic for interior design. The designer can use Jigsaw to ideate a design concept from client material (a), design the 2D and 3D mockups (b), and add music to enhance the presentation (c).

Ideating a design concept from client material. Zaha is an interior designer tasked with creating and presenting a redesigned interior for a client's new home. She received a photograph of interior from the client.

To begin, Zaha plans to create a design concept using the client's photo as a reference (Figure 7 a). She drags an Upload image puzzle piece from the Catalog Panel onto the Assembly Panel, and uploads the client's photo in the Input Panel. Zaha first identifies existing features in the client's home, such as built-in structures, furniture, and lights. Thus, Zaha uses the semantic search bar in the Catalog Panel to find a puzzle piece that can "identify the objects inside the image". Jigsaw returns the Tag image piece as the top result. She adds Tag image after Upload image to identify the objects. Zaha then uses the Ask GPT piece in ideation mode, with "contemporary interior concept" in the Task box, to assist her in brainstorming a concept. After clicking Run, Jigsaw has suggested a design concept of "An airy space with a minimalist fireplace and ladder, illuminated by low-hanging lamps."

Designing the 2D and 3D mockups. With a design concept in hand, Zaha proceeds to create the visual design (Figure 7 b). Zaha quickly duplicates the Upload image piece to start a new chain, using the client's photo as a reference again. She notices that the reference photo includes people, whom she wants to remove, and adds the Remove people piece. Zaha is interested in experimenting with AI image generation tools but acknowledges that the room's structure must remain intact for the design to be technically feasible. She uses the semantic search bar in the Catalog Panel to find a puzzle piece that can help "preserve the structure of the room". Jigsaw returns Get edge map and Get depth map as the top results. Zaha begins by testing the Get edge map piece and adds the Generate image from text and edge map piece, which takes in both image and text inputs. For text, she inputs the design concept suggested in the previous ideation chain.

Zaha feels that the generated image fails to retain the desired room structure. Recognizing this, she drags the pieces using edge maps into the trash bin. She tries Get depth map and Generate image from text and depth map instead, which better preserves the room's structure. To try different design variations, Zaha inspects the parameters tooltip for the Generate image from text and depth map model in the parameters sidebar. She discovers that she can tinker with the seed value to generate different variations.

Zaha is now satisfied with the redesign, except for the wooden ladder. She believes replacing it with a spiral glass staircase would better fit the contemporary concept. Thus, she searches for a puzzle piece that can modify an image using text instructions and finds the Generate image from text and image piece. Zaha instructs it to "replace the wooden ladder with a glass spiral staircase." The newly generated redesign now features a contemporary glass spiral staircase.

Zaha would like to visualize a 3D mockup. She discovers the Generate 3D model from image piece and attaches this to the chain, but finds that the generated results are low resolution. Thus, Zaha searches the Image section of the Catalog Panel for a piece that can help enhance the image. She finds the Increase image resolution piece.

Presenting the design to the client. To communicate the contemporary design concept, Zaha would like to incorporate a musical background to complement the design aesthetics. To achieve this, Zaha asks the Assembly Assistant to "help add music based on the image". The Assembly Assistant suggests the following chain of puzzle pieces: Caption image to understand the image, Ask GPT with ideation mode to brainstorm a fitting music description, and Generate music to generate the music (Figure 7 c). This outcome is a chill electronic music piece.

3.3.5 User Study

We conducted a user study to understand how Jigsaw could address designers' pain points in working with multiple foundation models, its potential to be integrated into design workflows, and identify improvement areas.

Participants and Procedure

We invited the ten designers from our formative interviews to participate in a one-hour remote user study. They were not exposed to Jigsaw's system or concept prior to the user study. Participants accessed Jigsaw through a web browser, shared their screen, and verbally explained what they were doing and thinking (think-aloud).

Introduction (10 minutes). Participants provided informed consent, and then received an introduction to Jigsaw's components, as described in Section 4.

Reproduction Task (15 minutes). Participants were asked to reproduce the interior design model mosaic described in Section 4.5. Participants used the starter image shown in Figure 7 a and a detailed design brief of the various steps they would need to create (see Section 4.5).

Free Creation Task (20 minutes). Participants were asked to freely explore Jigsaw and create their own model mosaics. We encouraged participants to build workflows beyond a simple chain and try out puzzle pieces involving multiple modalities.

Post-Study Interview (15 minutes). After the creation activities, we conducted a semi-structured interview asking about participants' experiences using Jigsaw, whether they could see Jigsaw being integrated into their design workflow, and to identify areas for improving the system.

3.3.6 Results and Discussion

All participants completed the reproduction and free creation tasks. Jigsaw appears to help designers discover and prototype new creative workflows. Designers suggested different future improvements.

Helping designers discover and utilize AI capabilities. Participants located AI capabilities via the Catalog Panel's semantic search bar or by filtering pieces by modality. Many participants mentioned that they were able to "discover new AI abilities [they were] previously unaware of (P9)". For example, P2, an illustrator, discovered the capabilities of ControlNet [56], a model that allows users to add additional control to a text-to-image model, such as a guiding sketch. Figure 8 shows the model mosaic created by P2, who used Jigsaw to create an audio-visual story. In Figure 8 a, she created visuals for her story. Instead of using a text-to-image model, she discovered and used ControlNet to generate images based on a starting sketch. In total, participants used 8 model puzzle pieces on average (μ =7.9, σ =1.29), and all participants explored beyond well-known models such as GPT and Stable Diffusion (see Appendix B for details). As the number of foundation models continues to increase, we plan to expand our set of puzzle pieces for designers over time.

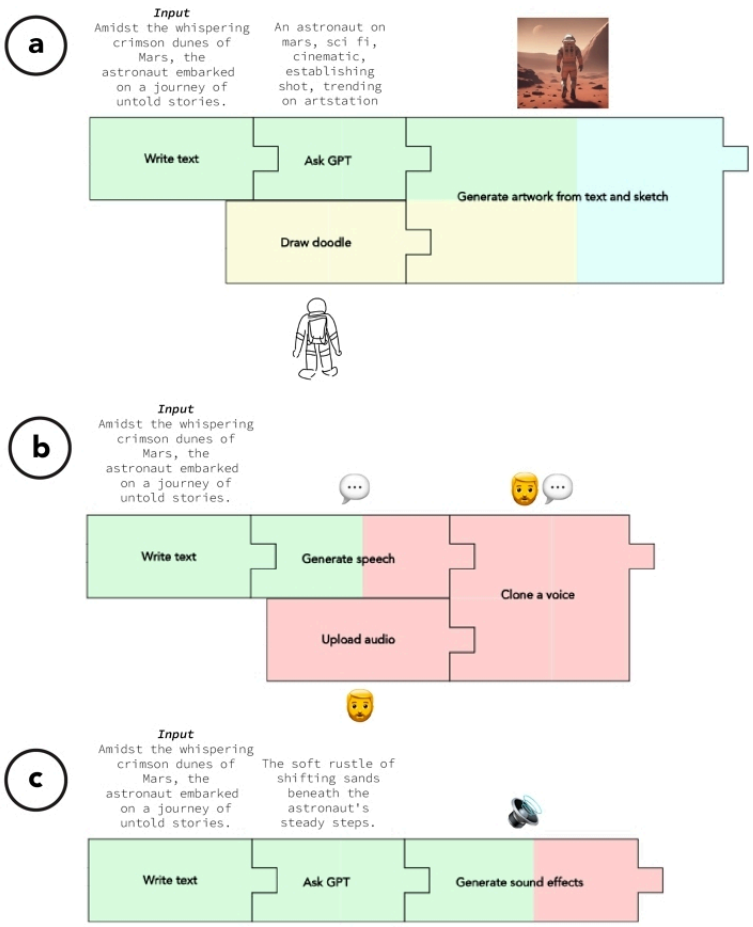

Figure X: Model mosaic by P2, an illustrator, to create an audio-visual story. P2 uses Jigsaw to create an illustration based on a text description and a reference sketch (a), generate narrations through a cloned voice (b), and generate accompanying sound effects (c).

Participants commented that tooltips "gave [them] a solid understanding of the capabilities of each of the puzzle pieces, like what can be expected as output and what types of inputs are suitable (P1)". In addition, participants expressed that the assisted prompting mechanism offered by the translation glue piece "allowed [them] to achieve satisfying results without the need to laboriously rephrase and tweak prompts (P2)": "Now, it's like I speak the AI's language. (P5)".

Supporting intuitive prototyping. Participants expressed that "[they] enjoyed the idea of building with AI visually with tangible puzzle pieces (P10)" and found it "easy to pick up and start designing (P3)". In particular, participants appreciated the error-proof design: "Knowing

which puzzle pieces can be connected expedites my prototyping. I see the same colors and receive snapping feedback. I don't spend time building a workflow and then compile it to find compatibility errors (P3)". A contribution of this research is showing how the design benefits of block-based VPIs, commonly catered towards novice programming learners [19, 35, 44], can be effectively applied to the realm of design prototyping for non-technical designers to work with AI capabilities. An interesting extension of Jigsaw could be the implementation of a "tutorial mode" to teach novice designers. The system would disassemble a model mosaic created by an experienced designer into pieces, allowing a novice designer to recreate it and learn from the experienced designer's design process.

Furthermore, participants mentioned that the ability to near-instantaneously see intermediate outputs in a chain "helped [them] to quickly test out ideas and make adjustments to individual steps as needed (P5)". This aligns with findings from prior research in interactive program debugging tools [22, 25, 37].

Serving as a brainstorming and documentation canvas. We observed participants making creative uses of the Assembly Panel, including using it to test different partial workflows before combining them into more complete workflows and using it to document their design explorations. Participants expressed that the Assembly Panel provides "a playground to be messy and experimental (P3)" and "makes it easy to track the evolution of an idea (P4)." Moreover, participants commented that the ideation glue piece was "helpful for brainstorming concepts at the beginning [of the design process] (P2)". We observed that designers occasionally passed the outputs of the ideation glue directly into a generation model, as shown in Figure 8 c. In other instances, designers maintained a shorter chain solely for concept generation. This is shown in the model mosaic created by P10, a game designer, in Figure 9, who created a video game character. He primarily used the short chain in Figure 9 a to generate concepts for his character. We observed that designers frequently created multiple chains to organize different stages of their design process. Participants noted that since the canvas

documents their creative process, it could serve as "a boundary object for sharing and explaining ideas to clients (P5)".

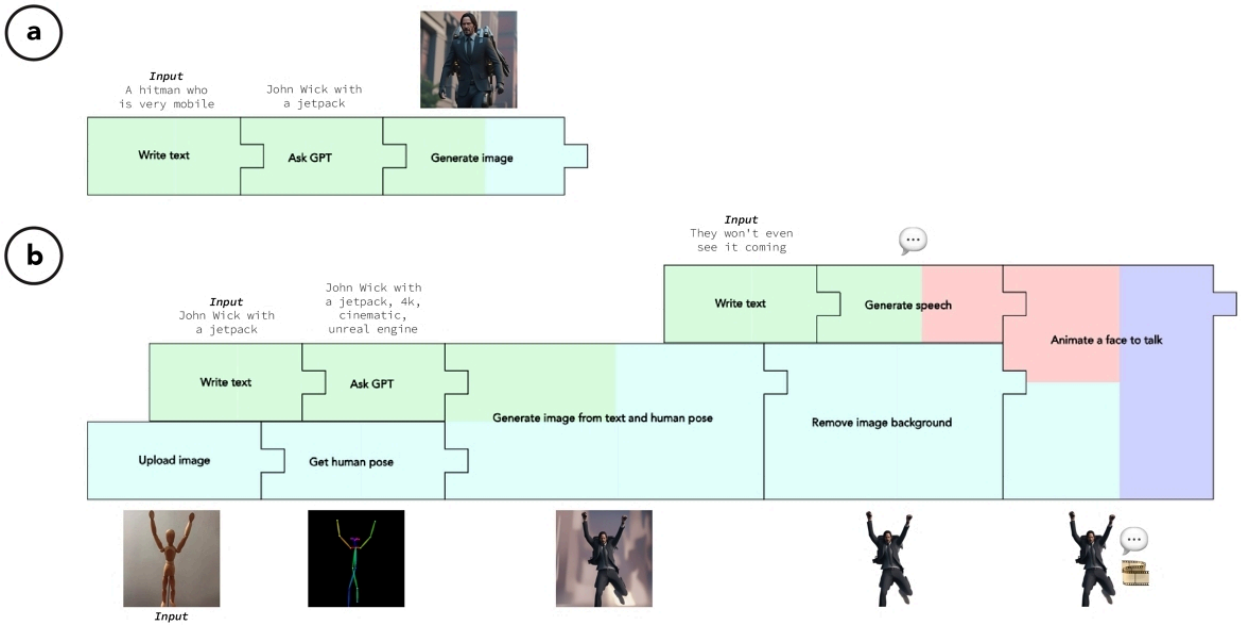

Figure X: Model mosaic by P10, a game designer, to create a video game character. P10 uses Jigsaw to create a character concept and preview the character's visuals (a). P10 then specifies a pose for the character and generates the character's visuals in the specified pose (b). P10 then generates a line for the character to say and animates the character to deliver the line.

Moreover, participants commented that the Assembly Assistant was useful in "generating an initial configuration of puzzle pieces to start working with (P1)". This aids in combating the "blank canvas syndrome (P6)", a common occurrence at the onset of a creative activity [23]. In Figure 9 b, P10 wanted his video game character to look like Superman flying. However, he initially struggled to come up with a method to accomplish this, so he sought assistance from the Assembly Assistant. The Assembly Assistant recommended a workflow of a reference pose image, a pose extraction model, and a ControlNet model that can be guided by pose. Given this workflow, P10 used a wooden mannequin to specify the pose for his character.

3.3.7 Limitations and Future Work

There are several avenues for improvement that we plan to address for future work. First, we currently implemented one AI model for each design task (e.g., Stable Diffusion for text-to-image). We plan to support the capability of switching between multiple alternative models. We will provide information on the tradeoffs between them (e.g., speed vs. quality) for both the user and as context for Jigsaw's subcomponents (i.e., semantic search and Assembly Assistant), facilitate easy side-by-side comparison, and allow users to filter models by certain criteria (e.g., text-to-image models with a typical runtime of under 10 seconds). Second, we are interested in expanding Jigsaw to let designers define custom puzzle pieces, as suggested by P10, and logic operators such as if/else statements and loops, common in other VPIs [12]. Third, we currently use LLMs as glue, using the text modality for intermediate reasoning (Section 4.1.4). We anticipate extending the glue piece to incorporate newer research on multimodal LLMs (MLLMs), such as GPT-4V [6] and LLaVA [29], to add information from additional modalities. Fourth, we plan to expand the Input and Output panels to handle real-time video and audio streams, as suggested by P9. Finally, participants noted that the Assembly Assistant was less robust to ambiguous tasks or tasks that require very complicated mosaics. As the Assembly Assistant uses GPT, we anticipate improvements to the Assembly Assistant as stronger versions of GPT are released. This is also a common challenge recognized by recent machine learning works that also aim to automatically combine expert models to solve complex AI tasks [21, 38, 47]. A path forward, as suggested by P7, could be to improve the Assembly Assistant to support back-and-forth interactions with the designer, becoming a design co-pilot that assists designers in creating complex workflows. As more designers use Jigsaw to create model mosaics, we plan to compile them into a template gallery that other designers can modify for their own use cases, as suggested by P8. We believe that the accumulated design templates can serve as a search space for the Assembly Assistant, enhancing its capabilities as a design search engine.

3.3.8 Summary

Jigsaw connects assembling *puzzle pieces* with *AI foundation model orchestration*, allowing designers to chain color-compatible AI puzzle pieces with no programming prerequisite. Jigsaw primarily improves the *extensible* characteristic of AI controllability (monolithic → modular building blocks) and shifts upwards in *intuitiveness* in the intuitiveness-controllability spectrum.

The three systems presented thus far demonstrate that the traditional trade-off between intuitiveness and controllability is not inevitable - mapping interface paradigms to AI control mechanisms creates tools accessible to non-experts. However, intuitiveness alone isn't sufficient for professional creative work. Professional designers need tools that fully maximize their domain expertise. The next section explores systems that transform professionals' domain knowledge and established practices into powerful AI steering mechanisms.

IV. Raising Controllability while Preserving Intuitiveness

Professional users often need more expressive control over AI behaviors than what conventional interfaces (e.g., text prompts or simple parameter sliders) offer. In this section, I prioritize controllablity - raising the ceiling for experts - by finding the right interface representations and control mechanisms that leverage the professional user's existing expertise and practices. PseudoClient builds on a designer's ability to recognize good design, letting the designer curate a handful of high-quality examples to train a personalized AI style model with meta-learning. Inkspire builds on designers' natural sketching workflow in product design, letting the designer use iterative sketching as the primary means to steer AI generation rather than prompt engineering. Together, these works demonstrate that developing AI control mechanisms grounded on existing expertise and practices can improve expressive controllability for professionals.

4.1 PseudoClient: Examples as an Interface

This chapter was adapted from my published paper: Lin, D. C. E., & Martelaro, N. (2021, June). Learning personal style from few examples. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (pp. 1566-1578).

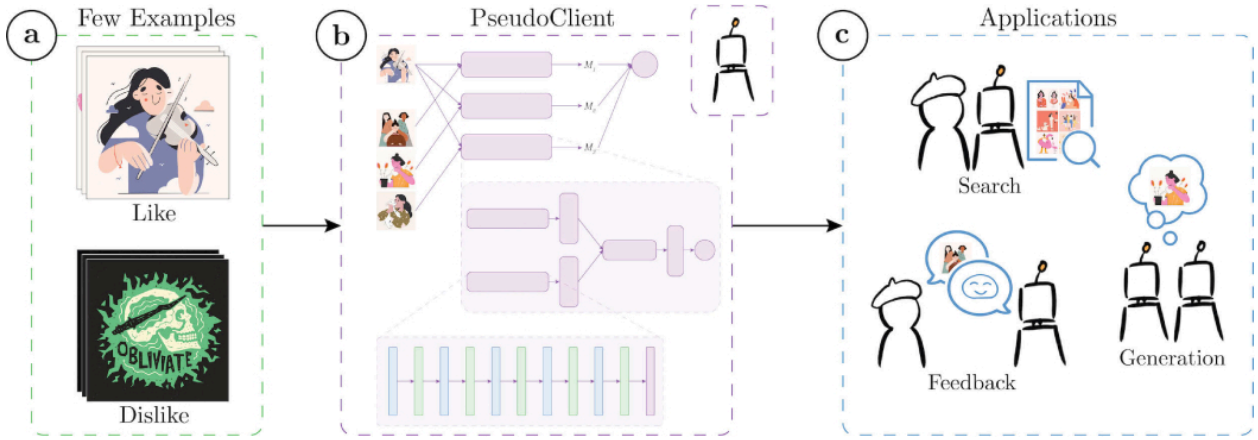

Figure X: Given a few design examples (a), PseudoClient learns a computational model of the client's personal style preferences (b) to support multiple practical design applications (c).

4.1.1 Introduction

A key role of the designer is being able to develop a firm grasp of the client's implicit tastes [4]. For example, in the case of graphic design, a hotel group may favor a minimal and luxurious feel, while an outdoor apparel company may prefer a more bold and rugged touch. However, since clients are rarely trained to constructively articulate their design ideas, their comments may often be vague and difficult to implement [56].

“Make it pop... is the dumbest thing you could say to a graphic designer.”
— Reddit user HylianHandy

To gain an understanding of the client's tastes, many designers arrange early meetings with clients to probe into their personal style preferences. Some designers use specialized tools to

help with this process of extracting implicit information from the clients, such as Brand Deck [2]. Brand Deck consists of 100 adjective cards (e.g., vibrant, futuristic, historic) that the designer asks the client to go through and sort into three piles: you are, you are not, and does not apply. One caveat of using a tool like Brand Deck is that designers will need to further examine whether the client's understanding of subjective words matches with theirs. Other designers prefer to give clients more visual examples, such as a logo book [14] or a mood board [34] consisting of a collage of different visual samples, and asking the client to pick a few samples that resonate with them. By using a selection of images, clients won't need to articulate their preferences through subjective words, but rather, by simply picking what they visually prefer. However, even when the client provides a pool of examples, recognizing a common pattern among numerous variables such as color, texture, and layout, synthesizing them into a composite understanding of the client's style preference, and using this understanding to generate a new design is a non-trivial task [15]. In this paper, we ask: can computational systems be used to aid in the task of learning peoples' style preferences from a set of example images?

Motivated by findings that high-level perceptions of a design are correlated with low-level features of appearance attributes [60], recent works seek to identify such low-level features using a computational approach to assist designers. Several works have explored extracting the underlying structure of designs such as the Document Object Model (DOM) tree of a web page [9, 31]. However, such methods only work with a small subset of design categories, such as webpages or vector images, where there is an encoded and explicit structure. For graphic design, where only bitmap information is available instead of structural elements such as shapes and text, other research has explored engineering various hand-crafted image features such as color histograms and edge maps [26] for predicting visual style. However, akin to the limitations of rule-based AI, hand-crafted features are limited in representation power and generalizability as some appearance attributes may be too complex or abstract to manually define. More recent works have experimented with using neural networks to extract a model of visual style due to their strong ability in automatically and adaptively learning the most relevant

low-level features [24, 59]. In our work, we look to use neural networks to automatically learn relevant features, but with two key distinctions. First, unlike most data-hungry machine-learning-based methods, our method only needs a small handful of examples. Second, prior work mostly attempts to categorize content into subjective style terms (e.g., futuristic, minimalist) [8, 59] based on the preferences of the crowd. Conversely, our work focuses on learning an individual's personal style preference, without ever having to put a word on it nor relying on generalized conceptions of design style.

We introduce a set of principles for learning a client's personal style from few examples. We then use these principles to develop PseudoClient, a deep learning framework that takes in a handful of examples and automatically learns a computational model of the client's personal style. Our use case focuses on graphic design, where we feed graphic design samples as input. We first ask the client to select a few examples they like and dislike. Using the samples provided, PseudoClient learns the client's personal style preferences using a metric learning approach [30] with twin Convolutional Neural Networks [29]. Exploiting the benefits of metric learning with deep neural networks, PseudoClient requires only a few examples, does not rely on limited hand-crafted features, and can be retrained quickly with additional samples. We conduct experiments to assess how PseudoClient compares to other methods as well as how it may be affected by various factors such as training sample size and the ratio of positive and negative samples. We find that PseudoClient outperforms our baseline methods and can be tailored for different needs. Finally, we discuss PseudoClient's capability of supporting the development of future design tools in three directions: search, feedback, and generation.

In summary, our contributions are three-fold:

- A set of principles for learning personal style from few examples. The principles are manifested in PseudoClient, a novel deep learning framework for learning an individual's graphic design style from a handful of examples and without relying on generalized conceptions of subjective style.

- Quantitative and qualitative experimental results demonstrating PseudoClient's advantages compared to other methods. We further examine how number of examples and ratio of positive and negative examples affect performance.
- A discussion of potential applications where PseudoClient can support augmenting designers' capabilities. We illustrate how PseudoClient is useful as a building block for multiple practical design applications by exploring three areas of design work.

4.1.2 Related Work

Our work is situated among literature in computational design understanding. More specifically, our research builds on prior work in two main branches: assessing aesthetics as a quality indicator and more fine-grained understanding of artistic style [46].

Assessing Aesthetic Quality

Computational methods of assessing the aesthetic quality of designs have a wide range of applications spanning from media editing to curation. Such methods typically involve defining a set of descriptors for some content and using them to predict human-labeled aesthetic ratings. Michailidou et al. [36] analyzes statistics of web page elements such as number of images, words, and links and Reinecke et al. [40] further incorporates page-level statistics such as number of leaves in a quadtree decomposition. In applications where the underlying structural information of the design is not available, such as bitmap photographs, prior works have engineered various visual features. Datta et al. [10] extracts 56 features based on photography concepts such as rule-of-thirds and depth-of-field. Isola et al. [25] investigates the correlation between image memorability and a set of high-level visual features such as object and scene semantics.

However, handcrafted features are nevertheless limited as some important aspects of a design may be too elusive or complex to manually define. Therefore, work in recent years has investigated the use of neural networks to automatically learn the most relevant low-level

features for pattern finding. For example, NIMA [50], one of the top performers on the Aesthetic Visual Analysis dataset [37], explores using CNNs to automatically discover important features without human supervision. In our work, we also exploit the strength of neural networks in feature extraction and do not use manual feature engineering. Rather than predicting an aesthetics rating, we predict a match score: how well does a design align with an individual's personal preferences as opposed to the weighted average of preferences from the crowd.

Understanding Style

Closely related to evaluating aesthetic quality is the task of understanding artistic style. We loosely categorize prior work in this domain into two approaches: tagging, where the task is to automatically assign labels to some content, and similarity, where the task is to learn the similarity across different content. Note that the two approaches are not necessarily mutually-exclusive.

Tagging has been extensively studied in information retrieval and recommender systems communities [11] although fewer works have explicitly approached tagging in terms of artistic style. Prior work in tagging by style has largely treated it as some form of classification problem. Shamir et al. [45] assigns paintings to painters and schools of art using image descriptors such as color histograms and edge statistics. Karayev et al. [26] predicts style labels such as bright and energetic for photographs and paintings using features such as color histogram and visual saliency. Wu et al. [55] models the brand personalities of mobile UIs with UI descriptors such as color, organization, and texture. Vaccaro et al. [51] predicts high-level fashion styles attributes such as tropical and exotic from low-level design element language such as color and material using polylingual topic modeling. More recent works have explored style tagging with deep neural networks. Zhao et al. [59] characterizes personalities for graphic designs such as cute and mysterious. Takagi et al. [49] proposes an expert-curated fashion style dataset containing 14 categories such as rock and street and found that modern computer vision classification networks are able to outperform fashion-naïve users but not fashion-savvy users. However,

since style tags are intrinsically subjective, labeling content with style tags is by nature a noisy task. Thus, our work does not aim to describe personal style with subjective style terms. Rather, our tags are simply whether a design “fits” or “does not fit” with an individual's personal style.

The question of “does a design fit or not fit with an individual's personal style?” can also be reformulated as “how similar is a design when compared to the individual's personal style?” This formulation opens up an interesting repertoire of research in “similarity by style” to draw inspiration from. D.Tour [41] and Webzeitgeist [31] allow search by stylistic similarity based on features of a web page such as DOM tree depth and number of leaf nodes. Several other works on learning similarity by style include applications in infographics [42], illustrations [18], icons [32], fonts [38], fashion [47], and 3D furniture models [33]. In our work, we borrow from the concept of similarity by style to define personal style based on similarity with a set of representative examples selected by the individual. Prior works have explored various methods for computing similarity, such as through color histograms [22] and Convolutional Neural Networks [54] (also see Comparison with Baselines subsection). Recent works in the Machine Learning community have shown the effectiveness of metric learning approaches [30, 35] for areas such as fashion [52] and home goods product design [6]. We build upon the successes of metric learning to design a metric learning framework for personal graphic design style. To guide our work, we distill a set of principles.

4.1.3 Design Goals

To support our objective of learning personal style preferences, we ground the development of PseudoClient in four central principles.

Principle 1: Learn by Example

Considering that design vocabulary (e.g., minimal, vintage) is inherently subjective and highly dependent and interpreted based on the individual's knowledge and experiences [46], we do not ask clients to indicate their preferences through such vocabulary nor do we attempt to fit their preferences into such vocabulary. Inspired by the mood board technique [34], we simply

ask the client to supply us with visual examples and learn to judge the client's personal style based on them.

Principle 2: Learn by a Handful

Since our task is to learn personal style, our examples come directly from the client of interest. However, asking the client to select massive amounts of examples is tedious. In addition, prior works have found that as the quantity of examples required from the client increases, the quality and consistency of the examples begin to decrease due to fatigue and the pressure to reach the target of providing a high number of samples [27]. Based on these observations, rather than being data-hungry, PseudoClient should be capable of working with only a handful of examples.

Principle 3: Learn by Juxtaposition

Findings from prior work in recommender systems [5] suggest that it is easier to select positive and negative samples than to discern likeability on a spectrum. Given this, we ask the client to provide us a set of examples they like (positive samples) and another set of examples they dislike (negative samples). Our task then is to determine whether a design fits the positive samples or the negative samples more closely. With only a limited number of examples to learn from, learning by juxtaposition allows us to formulate our problem of modeling the client's style preferences into more simplistic binary classification problems.

Principle 4: Learn by Multiple Comparisons

Studies in the learning sciences and cognitive psychology have observed that through comparison against multiple examples, one can quickly learn a common underlying structure, even if the individual examples are not fully understood [19, 20]. Building on this insight, we repeatedly do pairwise comparisons between the unseen design and the pool of reference examples provided by the client. For a more detailed description and diagram of the method, please refer to the Comparison Framework subsection.

4.1.4 System Implementation

Our four principles are manifested in PseudoClient and guide its implementation. To the best of our knowledge, we uniquely frame our task of modeling the client's personal graphic design style as a metric learning problem [30]. Given an unseen graphic design, our objective is to determine its similarity with a set of representative examples selected by the client. This allows us to then classify the design between two classes: client likes and client dislikes. The following outlines the implementation of PseudoClient, including (1) the Support Set, (2) the Comparison Framework, (3) the Juxtaposition Network, (4) the Embedding Network, and (5) the training setup. We wrote our learning framework in PyTorch version 1.6.0.

Support Set

We take inspiration from the mood board technique [34] by first asking the client to provide a small selection (Principle 2) of graphic design examples (Principle 1). More specifically, we ask the client to pick a few samples they like (positive samples) and another few samples they dislike (negative samples) (Principle 3). This is our support set S. We denote a subset of S containing only positive samples as the positive support set S+ and a subset of S containing only negative samples as the negative support set S−.

Comparison Framework

Figure 2 visualizes the Comparison Framework. Given an unseen graphic design It and a positive support set S+, we perform pairwise comparisons (Principle 4) between It and each reference sample Ir1, ..., rn ∈ S+ to compute their respective match scores M1, ..., n through our Juxtaposition Network (see Juxtaposition Network subsection on how M is computed). A high match score means that It is predicted to be in the same class as Ir. This implies that the unseen design It matches the client's personal style, since Ir is a sample that the client likes. Conversely, a low match score means that It is predicted to be in a different class as Ir. This implies that the unseen design It does not match the client's personal style. We are therefore learning to classify designs indirectly by evaluating their similarities with a set of labeled referencing examples (positive support set S+). We compute match scores with the positive support set S+ as opposed

to the full support set S based on our findings from empirical pilot testing. We found that the client's negative examples tend to be less consistent, often consisting of many diverging styles the client dislikes. We then take the median of the pairwise match scores M1, ..., n as the overall predicted match score ^M. We take the median rather than the mean to decrease the effects of outliers.

^M(It)=MedianIr∈S+(M(It,Ir))

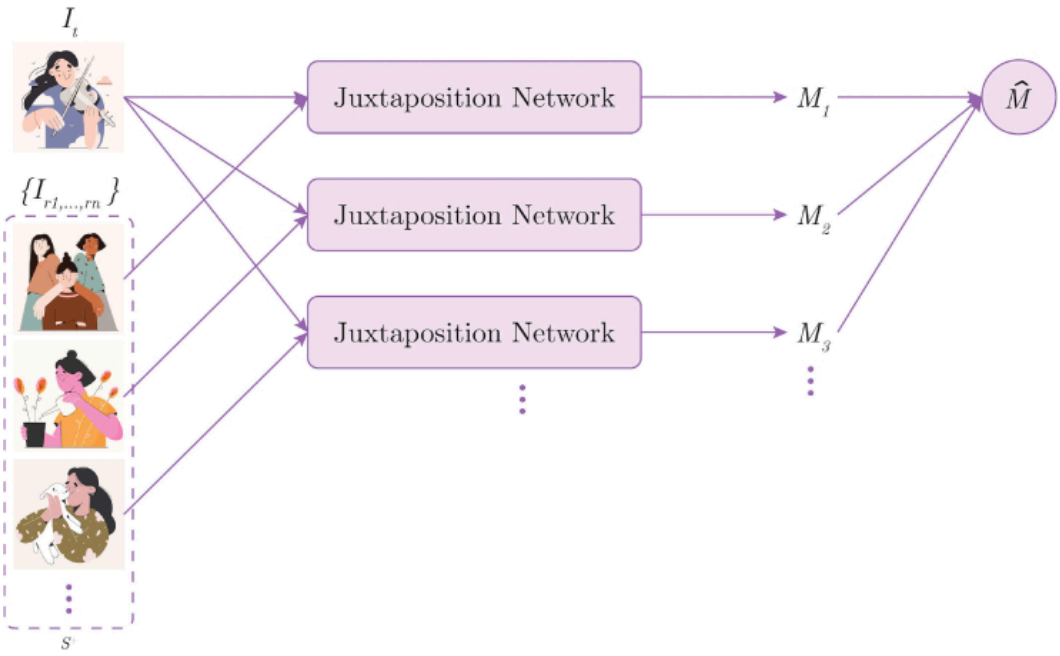

Figure X: The Comparison Framework. We compute pairwise match scores between the test image and each reference image in the positive support set. The final match score is the median.

Juxtaposition Network

Since we want to work with a small support set (Principle 2), we approach the challenge of learning a model to accurately predict the match score M as a few-shot learning task [17]. Prior work by Melekhov et al. [35] in image matching showed the strength of Siamese Networks [29] in generalizing from small datasets. We build upon this and learn the similarity function between two graphic design inputs. We design a Juxtaposition Network resembling a Siamese Network with twin Convolutional Neural Networks (CNN).

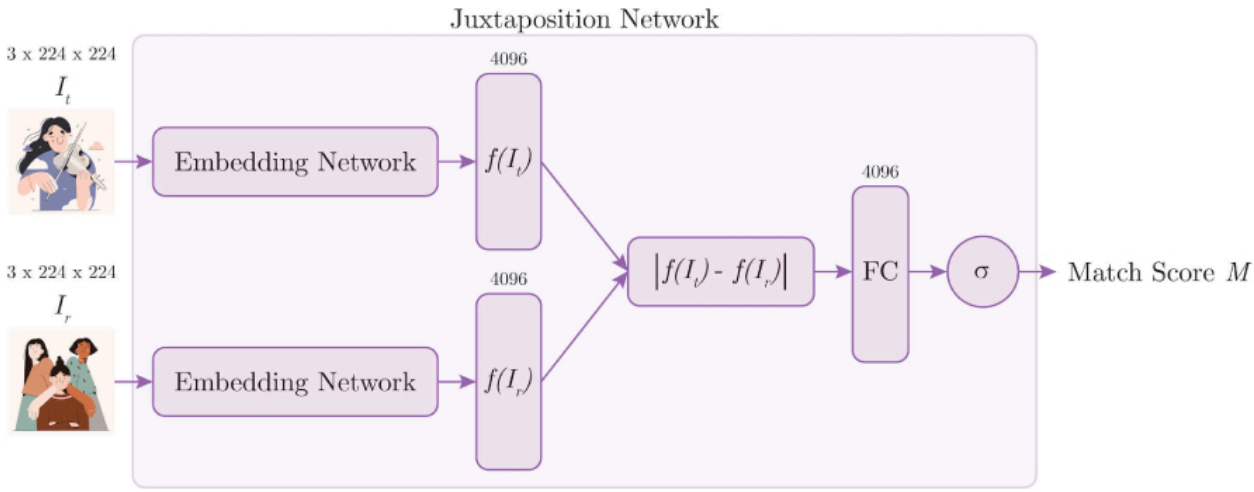

Figure X: Architecture of the Juxtaposition Network.

Figure 3 visualizes the architecture of the Juxtaposition Network. The Juxtaposition Network takes in a test image It and a reference image Ir ∈ S+ as inputs and computes a match score M with range = [0, 1] as output. Our Juxtaposition Network uses a twin architecture. We first encode each 3x224x224 RGB input image It and Ir into 4096-dimensional feature embeddings f(It) and f(Ir) through twin Embedding Networks (see Embedding Network subsection). We then compute the weighted L1-distance D between the two feature embeddings.

$$D(It,Ir)=|f(It)-f(Ir)|$$

We then translate the distance D into a match score M, which is the probability that the two inputs belong to the same class, by passing it through a fully-connected layer (FC) with learned weights W and a sigmoid activation function (σ).

$$M(It,Ir)=11+\exp-W \cdot D(It,Ir)$$

Embedding Network

Figure 4 visualizes the architecture of the Embedding Network. The Embedding Network takes in a 3x224x224 RGB image as input and extracts a 4096-dimensional feature embedding as

output. The feature embedding is a vector representation that captures visual features of the image. We use five convolutional blocks (a 3x3 convolutional layer and a 2x2 max-pooling layer) and two fully-connected layers. We apply batch normalization and ReLU activation after the convolutional layers and sigmoid activation after the fully-connected layers. Our architectural design resembles the well-researched VGG architecture [48] and performs well empirically. While there are certainly techniques to further optimize performance [16], they are not the focus of this paper.

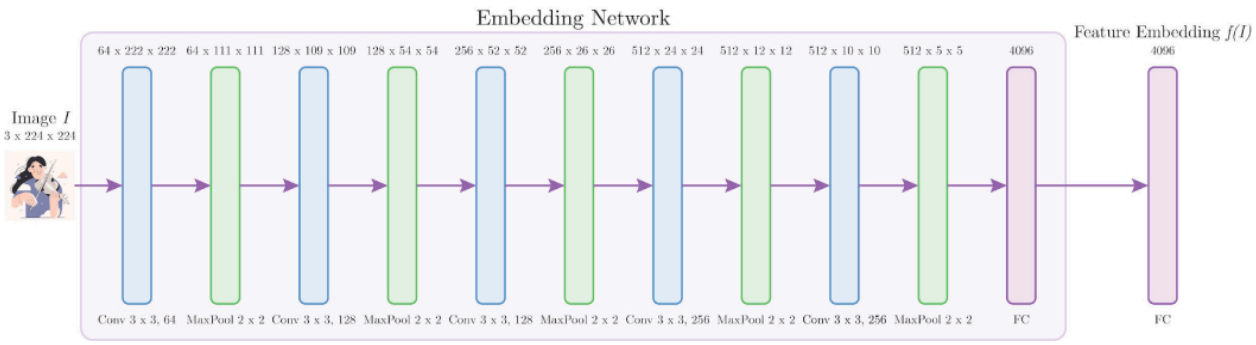

Figure X: Architecture of the Embedding Network.

Training Setup

Our training setup consists of two stages: pre-training and finetuning.

Pre-training. To allow our model to quickly learn a client's personal style with few examples, we first pre-train a network on a dataset of graphic design images collected from dribbble.com. The dataset consists of six different classes (flat, geometric, line, minimal, pop, vintage) with 50 images each, yielding a collection of 300 images. We collected the dataset based on querying relevant keywords, with class labels self-tagged by the artists. Note that the main purpose of pre-training is to seed the network, letting our model first learn some basic visual features of graphic design, such as edges, patterns, or general shapes, an interesting property of neural networks [58]. Thus, our focus is not to bin all graphic design work into these six subjective categories. Rather, when the client trains PseudoClient to model their own tastes, instead of training a model from scratch with completely random initialized weights, the client's model can

have a head start by building on the pre-trained model via transfer learning [39], resulting in a much shorter training time.

We process the dataset into pairs of graphic design images with an assigned binary label y. If the images in a pair belong to the same class, we set y = 1. If the images in a pair belong to distinct classes, we set y = 0. The pairs are randomly sampled. We resize the images into 3x224x224 pixels with RGB channels and normalize each color channel with mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225] based on the statistical distribution of ImageNet [12]. We split our training and validation sets with a 9:1 ratio. We use the Adam optimizer [28], a batch size of 32, a learning rate of 1×10^{-5}, and train for 50 epochs. Since the labels are binary, we use a cross-entropy loss L for every training batch B.

$$L(B)=\sum_{It,Ir,y \in B} y\log(M(It,Ir))+(1-y)\log(1-M(It,Ir))$$

Training takes around an hour to complete on an NVIDIA GeForce RTX 2080 Ti graphics card with 11GB of memory.

Fine-tuning. This is the stage where the model learns the client's personal preferences. Given a pre-trained model, we fine-tune the model so that it reflects their own tastes. We begin with the pre-trained weights and train the model with respect to their support set S as the new training dataset. Data processing procedures are the same as for the pre-trained model. Therefore, we learn by juxtaposition (Principle 3), where y = 1 means that the graphic designs in the pair are both liked or disliked by the client (same class) and y = 0 means that one of the graphic designs in the pair is liked by the client while the other is disliked by the client (different class). This means we are not directly learning whether a design is liked or disliked by the client, but rather learning its similarity with both the liked and disliked example sets. We train our model until it can consistently predict the correct y label for each pair. We use the same optimizer and loss function as the pre-trained model, a batch size of 16, a learning rate of 1 ×

10^{-8}, and train for 20 epochs. Training takes around ten minutes to complete on the same hardware as for the pre-trained model.

4.1.5 Experiments

To evaluate the effectiveness of PseudoClient, we performed several experiments. This serves as a litmus test of the system's performance. The following outlines our setup and various experiments, including:

Comparisons with baselines. We evaluate PseudoClient's performance by comparing against other methods, namely, softmax-based approaches (Convolutional Neural Networks) and traditional distance measurements (color histogram distance).

Exploration of various factors. To discover usage guidelines for future designers and offer an understanding of how designers can work with PseudoClient for their needs, we explore how various factors can affect PseudoClient's ability to learn personal style. We focused on two factors: number of examples (dataset size) and different ratios of positive and negative examples (class imbalance).

Query results. Finally, as graphic design is a highly visual medium, we want to see how PseudoClient performs qualitatively to uncover insights that may not be portrayed through numbers. We do this through a simple image retrieval task, visualizing the retrieval results.

Table 1: The accuracies of different methods between participants P1 – P5 and overall. The highest accuracy is highlighted in bold.

Method	P1 Accuracy	P2 Accuracy	P3 Accuracy	P4 Accuracy	P5 Accuracy	Overall Accuracy
Chance	50.00%	50.00%	50.00%	50.00%	50.00%	50.00%
Color Histogram	69.00%	26.00%	53.00%	55.00%	58.00%	52.20%
CNN	67.00%	63.00%	76.00%	53.00%	62.00%	64.20%
PseudoClient (Ours)	86.00%	77.00%	90.00%	64.00%	80.00%	79.40%

Setup

Given that our task is to learn personal style preferences, we evaluate our accuracy in doing so with five people: two of the paper's authors and three volunteers. For each participant, we ask them to first come up with a simple design idea (e.g., an illustration for a local French bakery). We then ask them to select 70 images that fit their design idea (positive set) from dribbble.com. This is for our experimental purposes, such that we have enough images for various levels of training and testing. In an actual usage scenario, participants would not need to supply as many examples (see Number of Examples subsubsection).

After all participants finish selecting their positive sets, participants then select another person's positive set (from the pool of positive sets of all other participants) as their negative set. Our requirement is that their selected negative set doesn't fit their design idea. One potential constraint of this approach is that a participant may struggle in selecting a negative set from the pool if all other positive sets match closely with theirs. However, such an issue did not occur during our specific study (maybe because participants had different design ideas). Thus, each positive set was stylistically different from the negative set. For a snapshot of what the participants picked as their positive and negative sets, please refer to Figure 5. For each positive and negative set of 70 images, we randomly allocate 50 as our test set and 20 as our training set. This means that 50 of the 70 images will be used as a true test set and not be used for training. For each training set of 20 examples, we further randomly sample various training sets of 1, 5, 10, and 20 examples for our later study on how different numbers of training samples affect performance (see Exploration of Various Factors subsection). For our comparison with baselines and qualitative query results studies, we use the training set of 5 examples (i.e., training with only 5 positive and negative examples). This follows our objective of learning by a handful of examples (Principle 2).

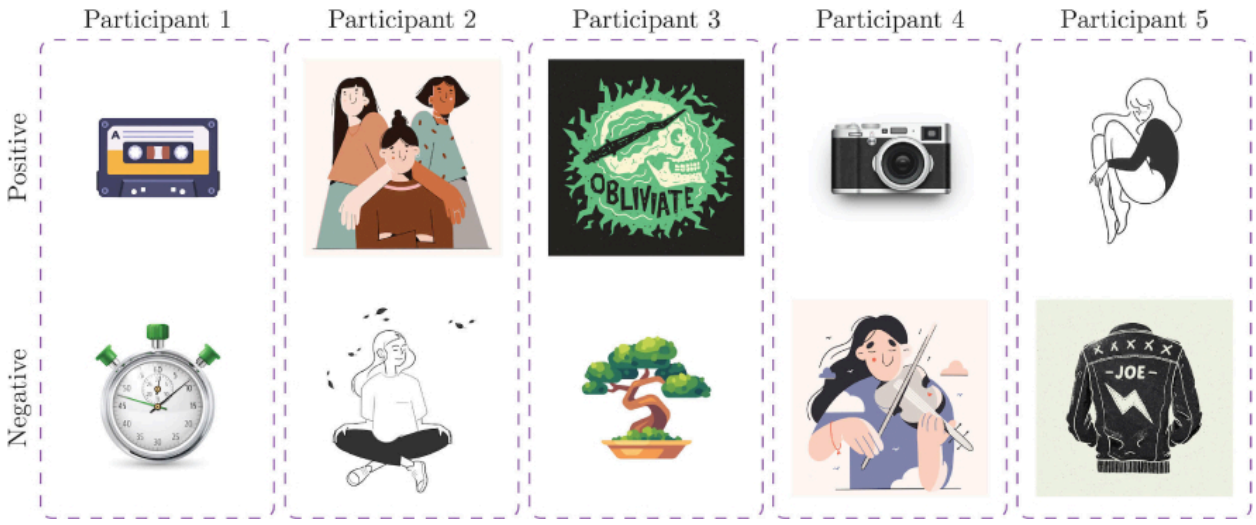

Figure X: A snapshot of each participants' positive and negative sets. Each column represents a participant. The top row displays an example of their positive set and the bottom row displays an example of their negative set.

We train a separate model for each participant. To measure the accuracy of the model, we then compute a match score for each of the 50 examples in their unseen positive test set as well as for each of the 50 examples in their unseen negative test set, by pairwise comparisons with all examples in their positive training set (see section:comparison-framework subsection). Note that the match score M is a numeric value with range = [0, 1] where a higher value implies higher predicted similarity with the positive support set, and a lower value implies a lower predicted similarity with the positive support set. For the positive test set, if M is greater than the threshold of 0.5, we note down a correct true positive (TP) prediction. For the negative test set, if M is less than the threshold of 0.5, we note down a correct true negative (TN) prediction. Our accuracy is then given by numberofTP+numberofTN100. Finally, we take the average of the accuracies for each participant as the overall accuracy.

Comparison with Baselines

We implemented two baseline models to compare against our method: (1) color histogram and (2) a convolutional neural network (CNN). Our first baseline is based on the distance of color histograms, which is a widely used metric for evaluating similarity between images [22]. We first extract a 3D histogram from each RGB image, using 8 bins per channel and normalize with range

= [0, 256]. We then flatten the histogram, yielding a 512-dimensional vector. To compute the distance between a pair of vectors, we compute its correlation [3]. The correlation is a numeric value with range = [0, 1], where higher correlation implies smaller distance and more similarity and and lower correlation implies larger distance and less similarity. We compute the overall accuracy of the color histogram method using a similar procedure as the second paragraph of the Setup subsection. However, instead of using a fixed threshold of 0.5, we set the threshold as the mean correlation value between images in the training set since the distribution of correlation values differ greatly across different participants.

Our second baseline is a standard implementation of a CNN, representing a typical neural-network-based approach for classification. We use the architecture of our Embedding Network with a single output node. Rather than computing a match score, we directly classify whether the unseen test image is liked or disliked by the participant. The accuracy is then given by numberofcorrectpredictions100 and we also take the average of the accuracies for each participant as the overall accuracy.

Table 2: The accuracies of PseudoClient when given different numbers of examples between participants P1 – P5 and overall. The highest accuracy is highlighted in bold.

Examples P1 Accuracy P2 Accuracy P3 Accuracy P4 Accuracy P5 Accuracy Overall Accuracy

1 47.00%74.00%47.00%53.00%70.00%58.20%

5 86.00%77.00%90.00%64.00%80.00%79.40%

10 83.00%73.00%93.00%66.00%75.00%78.00%

20 88.00%80.00%96.00%77.00%92.00%86.60%

Table 3: The overall true positive percentages, true negative percentages, accuracies, and F1 scores of PseudoClient when given different ratios of positive and negative examples. The highest results are highlighted in bold.

Number of Positive:Negative Examples	True Positives	True Negatives	Accuracy	F1 Score
10:5	82.40%	68.40%	75.40%	0.77
5:10	65.20%	82.80%	74.00%	0.71

By learning to classify designs indirectly via learning their similarities with a positive support set, our hypothesis is that PseudoClient would perform better than a standard CNN implementation, given the nature of our task: “whether a design is more similar to the set of like examples or the set of dislike examples” seems more learnable than naively judging “whether a design is liked or disliked.” The latter is an intrinsically ambiguous task, since like/dislike more often falls on a scale as opposed to being a perfect dichotomy. In addition, we also hypothesize that color alone would not be sufficient for determining personal style. From Table 1, we observe that color histogram performs only marginally better than random chance. The CNN, which is essentially PseudoClient's Embedding Network, still struggles to classify consistently. Overall, we observe that PseudoClient is able to outperform our two baselines for all participants, supporting our hypotheses.

Exploration of Various Factors

5.3.1 Number of Examples. We first investigate how supplying different training sample sizes affects PseudoClient's performance. Our hypothesis is that accuracy will increase as the number of examples increases, and we test our hypothesis by training separate models using 5, 10, and 20 positive and negative examples and evaluating their accuracies. We do not explore beyond 20 examples since it goes beyond the definition of “few examples” based on feedback from our participants. We also evaluate how well our pre-trained model can generalize to personal style without any further fine-tuning and given only 1 reference example.

Table 2 summarizes the accuracies of PseudoClient when given different numbers of examples for each participant and overall. We observe that accuracy generally increases as the number of examples increases, confirming our hypothesis. However, interestingly, when compared to the

overall accuracy of the models using 5 examples, the overall accuracy of the models using 10 examples did not increase and even dipped slightly. This may suggest that an increase in performance may only be prompted by a sufficiently large increase in sample size. Another interesting observation is that our pre-trained model, given only 1 guiding example, is able to perform better than random chance by some amount for P2 and P5. The reason may be that the examples chosen by these participants are more uniform, granting the single example a greater representative capacity. Hence, designers should note that clients who have tighter style preferences may not need to supply as many examples.

Ratio of Positive and Negative Examples. Previously, we trained our models with equal amounts of positive and negative examples. We thought it would be interesting to also investigate how performance would be affected if the client gives more positive examples and fewer negative examples, and vice versa. We hypothesize a higher TP when given a higher ratio of positive examples and a higher TN when given a higher ratio of negative examples. We experimented with two setups: 10 positive examples with 5 negative examples and 5 positive examples with 10 negative examples.

Table 3 summarizes the overall true positive percentages, true negative percentages, accuracies, and F1 scores of PseudoClient when given different ratios of positive and negative examples. We observe that the model predicts true positives more consistently when given more positive examples and predicts true negatives more consistently when given more negative examples, supporting our hypothesis. This reveals an interesting property: designers may adjust the ratio of positive and negative examples required for their specific goals. For example, if the goal is targeted towards identifying and filtering out what the client dislikes, then the designer may ask the client to focus on supplying more negative examples.

Query Results

We qualitatively evaluate the performance of PseudoClient with an image retrieval task. We query for the top 10 images with the highest match scores from a database of graphic design

samples. Our database consists of a subset of the Dribbble dataset from [8] and some examples selected by the participants. None of the samples were used for training.

Figure 6 visualizes two example queries. The samples bounded by the dashed lines are the positive examples used for training (positive support set) and the samples bounded by the solid lines are the top-10 queried results ranked from 1 to 10. We observe that PseudoClient is able to retrieve stylistically similar graphic designs by synthesizing from a few examples. Interestingly, a couple retrieved designs in Figure 6 a even belong to the same artist as the provided examples, demonstrating PseudoClient's capability of recognizing personal design style. Note that the retrieved samples don't necessarily have similar color as the provided examples. For example, the top-ranking retrieval in Figure 6 b has a light cream background despite most of the examples having darker backgrounds. Nonetheless, its resemblance to the examples in terms of artistic style is apparent. One limitation we discovered from the queried results is that PseudoClient judges samples in a more holistic sense and may overlook fine-grained but important details such as font styling (see Figure 6 b). For example, serif and san-serif fonts may elicit different emotions [44]. We suggest an approach to address this in the limitations and Future work section.

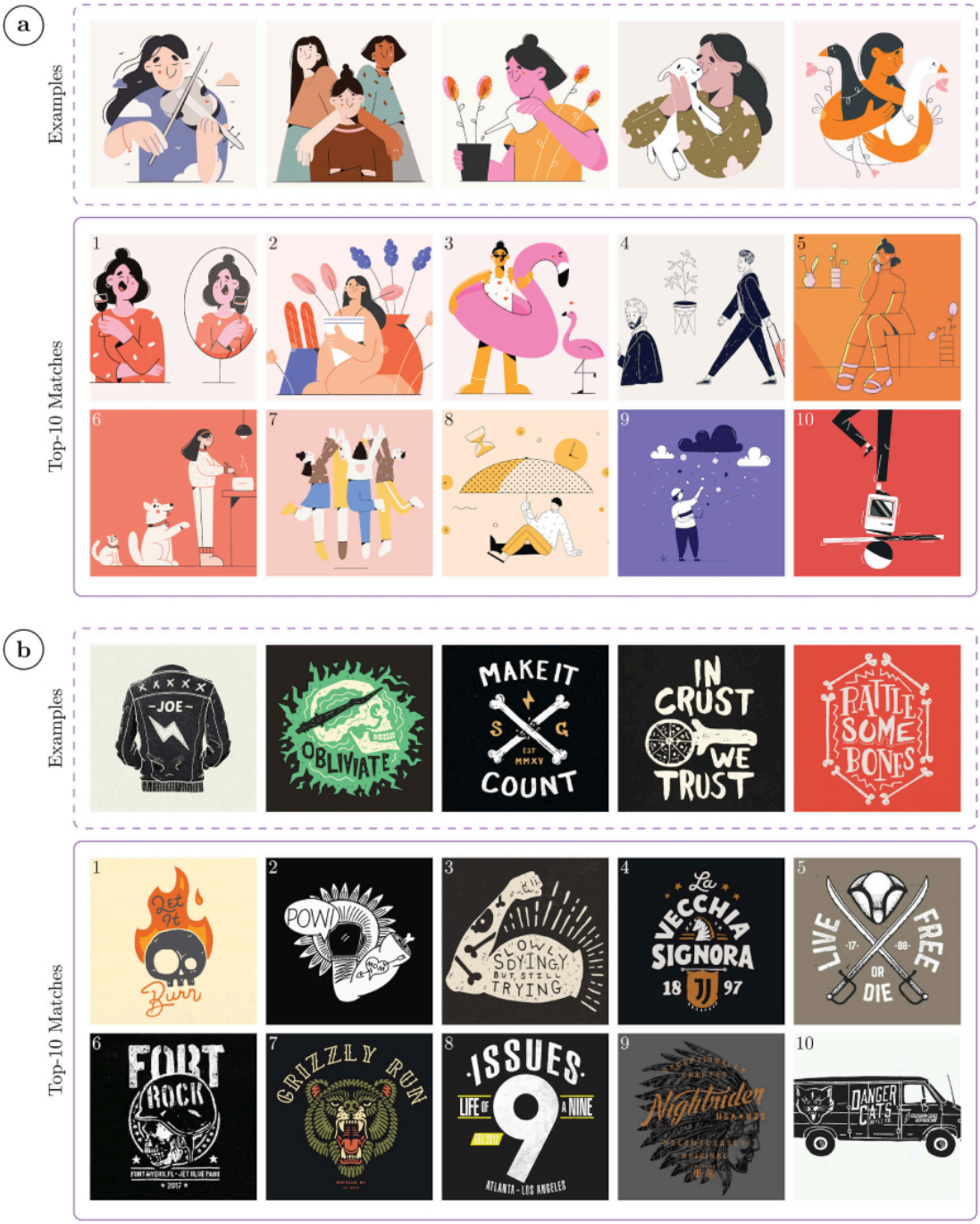

Figure X: (a) and (b) show two example query results. The dashed lines contain the positive examples used for training (positive support set). The solid lines contain the queried samples with the highest match scores, ranked from 1 to 10.

4.1.6 Applications

We suggest possible applications that can be enabled using PseudoClient by illustrating its use cases in augmenting designers from three directions: (1) search, (2) feedback, and (3) generation.

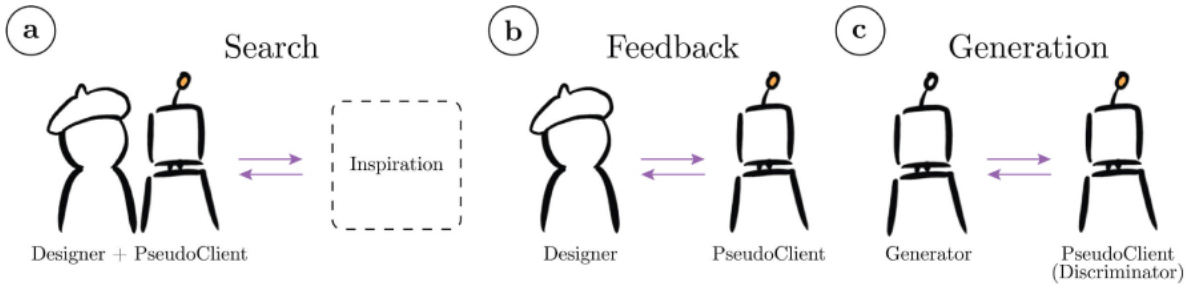

Figure X: PseudoClient can be used as a building block for multiple practical design applications. We explore applications from three directions: (a) search, (b) feedback, and (c) generation.

Search

A natural application of PseudoClient is an example-based, personalized style search engine (Figure 7 a). Prior work has shown that being able to find high quality examples is a crucial part of the creative design process for not only gaining inspiration, but also exploring alternatives and performing comparative evaluations [23]. An example usage scenario may be that shown Figure 6. The designer may first request a few examples from the client. Using these examples, the designer may then search for even more examples of similar style. Note that unlike existing search tools built into many design sharing websites, a search engine built on top of PseudoClient has the benefit of searching directly with examples instead of tagging based search with subjective keywords [8]. Furthermore, style-based search can surface designs that have not been tagged with keywords. This setup provides a powerful mechanism for locating potential relevant design examples to draw inspiration from, without having to overwhelm the client with the task of providing large quantities of examples themselves or going through lengthy meetings for designers to probe and synthesize their tastes.

Extending beyond searching for design examples, PseudoClient may also be used for clients to search for designers. Since designers often also have their own personal style, it is sensible for clients to find designers who have personal styles that match their tastes. Given a designer's design portfolio, PseudoClient can assess the portfolio's overall similarity with the client's personal style preferences. The client may thus do a “reverse designer search” with PseudoClient to discover the top designers who most fit their needs. Similarly, one may also do a “reverse design community search” with PseudoClient to discover the top design communities that are most well aligned with their personal style preferences. A fellow designer pointed out that being able to cluster designers or design communities by style may also reveal how designers are influenced by one another and unravel interesting design trends and patterns.

Feedback

The ability of PseudoClient to assess similarity by style opens the possibility of an automatic design feedback system. By referencing the few examples given by the client, PseudoClient can provide rapid, automatic design feedback to the designer for evaluating how well the designer's design drafts align with the client's personal tastes (Figure 7 b). Such an application can potentially increase the quality of design work by allowing the designer to receive timely critique for iterative exploration of alternatives and ensure that their design direction remains aligned with the client's tastes, without the constraint of the client's limited availability [13]. This is especially timely as an increasing amount of design work shifts to nowhere and everywhere (remote) where latency between design cycles is substantially magnified due to time zone differences and the absence of face-to-face meetings.

Another interesting use case of PseudoClient's capability of giving feedback, suggested by another fellow designer, is a tutorial system for beginners to mimic the styles of masters.

“Art is sourced. Apprentices graze in the field of culture.”
— Jonathan Lethem, Writer

A common way for beginner artists to improve their techniques is by performing master studies. During this process, the beginner studies the techniques of a master by recreating a piece of work using a similar style [1]. However, whether the master copy truly aligns with the master's style may ultimately be difficult to determine. Given a handful of the master's work as examples, PseudoClient can be applied as a simple tutorial system by rating the beginner's master copy. Implementing an activation map to reveal which regions of the copy matches with the master's style may further increase explainability and actionability of PseudoClient's feedback [7]. Such an application can also extend to mimicking the artist style of an era or genre.

Since PseudoClient effectively learns a model of one's personal style preferences, its feedback capability can also be repurposed for personal authentication. Google's reCAPTCHA system serves hundreds of millions of CAPTCHAs every day to tell humans apart from computers [53]. Users are presented a set of 9 or 16 square images and asked to identify which images contain certain objects. For our use case, we can first ask a user to select a small positive support set of design images upon the creation of an account. PseudoClient may then serve as an authentication system, in similar fashion to reCAPTCHA, by asking the user to select the design images that most align with their personal style preferences and judging its overall stylistic similarity with the positive support set associated with the account for authentication. The core assumption behind is that one's unique style preference can be personal enough to serve as a mental metric (as opposed to biometrics) for personal identification.

Generation

PseudoClient may also serve as a key component in generative design methods, such as Generative Adversarial Networks (GANs) [21]. We can think of GANs as two actors, a generator and a discriminator, working against each other to generate realistic designs. The generator attempts to hallucinate “fake” yet plausible designs. The discriminator attempts to differentiate between “fake” designs, generated by the generator, and “real” designs, from a set of real design examples. Competing against each other, each actor becomes better and better at doing

its job, until the generated “fake” design is barely distinguishable from a “real” design. PseudoClient may be used as a discriminator (Figure 7 c). Instead of differentiating between “real” and “fake” designs, our new discriminator differentiates between designs that the client “likes” or “dislikes” with respect to their personal style preference. On the other hand, the generator (or “PseudoDesigner”) attempts to minimize the difference between its generated designs and the “like” examples provided by the client. Designers may thus utilize this adversarial behavior to synthesize designs that resemble the client's personal style preferences. We hope that such a generative system can become a useful tool, not to replace the role of designers, but to assist designers in serving as a source of inspiration, for rapidly prototyping new alternatives, and fundamentally making design work more intuitive to novices.

4.1.7 Limitations and Future Work

While PseudoClient performs well and appears to learn personal style, there are several limitations. These limitations suggest interesting avenues for future work. First, while PseudoClient is able to learn personal style holistically, more fine-grained elements such as variations in typeface may be overlooked, perhaps due to downsampling. A possible approach to address this may be to first semantically segment a design in order to distinguish between different design elements, as opposed to treating the entire design as a whole. For example, a design's typeface or background (with foreground subtracted) could be treated as separate features for learning. This being said, adding more feature engineering could lead to overly constrained systems. Second, a limitation raised by a fellow designer is that while PseudoClient is able to learn style in a visual sense, it is not able to explicitly understand a design based its content. For example, the use of skulls and bones in Figure 6 b may correlate with specific styles (e.g., grunge, vintage, spooky), although their usage per individual may be different based on how they interpret these symbols and the context that surrounds them. It would be interesting to explore how content could correlate to personal elicitations of style and how (in)consistent they might be across different individuals. Third, while our evidence suggests that PseudoClient is able to distinguish between different styles, we may see that the positive and negative styles of participants in our experiments were quite different from each other. This motivates future

work on exploring various degrees of similarity between positive and negative styles, such as investigating how PseudoClient can learn very subtle style differences. Finally, PseudoClient currently functions more as a component rather than as a fully fleshed out design application. For future work, we plan to work with PseudoClient as a design material [57] to implement some of the design tools discussed in the Applications section and evaluate them via user studies with designers and clients.

4.1.8 Summary

PseudoClient utilizes *example-based* interaction with *few-shot meta-learning*, enabling graphic designers to teach AI models their personal style through just a handful of examples. PseudoClient primarily improves the *personalized* characteristic of AI controllability (generic → personalized) and shifts upwards in *controllability* in the intuitiveness-controllability spectrum. PseudoClient reduces steering from hours of model training to minutes. The next chapter pushes responsiveness to seconds.

4.2 Inkspire: Scaffolded Sketching as an Interface

This chapter was adapted from my published paper: Lin, D. C. E., Kang, H. B., Martelaro, N., Kittur, A., Chen, Y. Y., & Hong, M. K. (2025, April). Inkspire: supporting design exploration with generative AI through analogical sketching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (pp. 1-18).

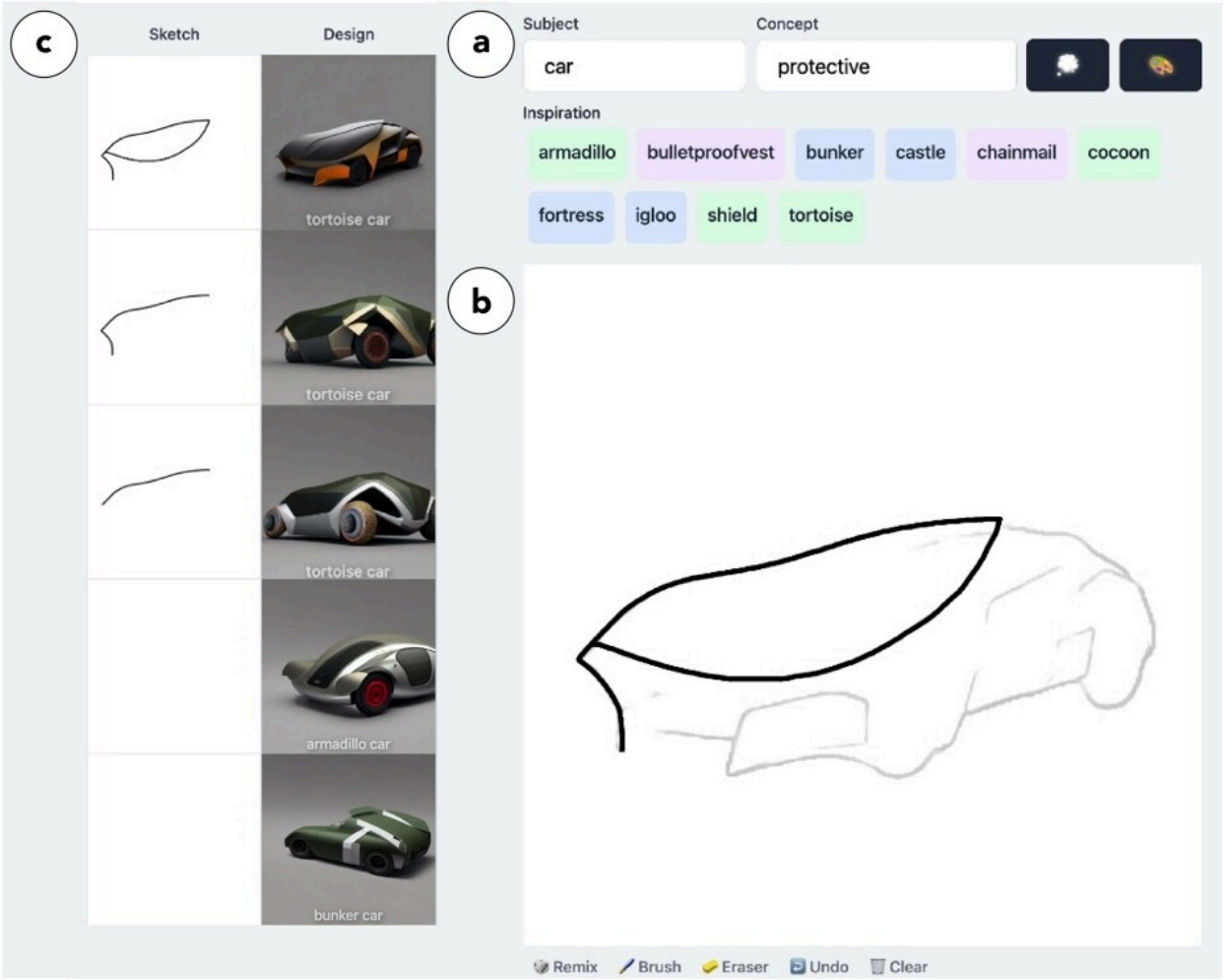

Figure X: The Inkspire interface. The designer may use the Analogical Panel (a) to ideate analogical inspirations for abstract concepts (e.g., "protective car" → "tortoise car"). The designer may sketch on the Sketching Panel (b) to iteratively guide AI design generations. For each iteration, we display a sketch scaffolding under the canvas. This scaffolding is created through abstracting AI designs into lower fidelity. Finally, the designer may view the history of iterations on the Evolution Panel (c).

4.2.1 Introduction

We have seen significant progress in the capabilities of text-to-image (T2I) models, many of which are now able to generate realistic images using text [9]. These models not only accelerate the process of converting thoughts into visuals but can also potentially create serendipitous inspirations for users [25]. Recent research has also opened up new possibilities for translating one image representation into another, such as transforming a sketched drawing into detailed designs [72]. Consequently, many designers have begun embracing the use of T2I models to enhance their creative work.

However, despite the proposed benefits T2I models, integrating them into the designer's creative workflow can be challenging. In particular, recent works have observed that designers experience a high level of fixation [35] when using generative AI [63] leading them to explore fewer novel ideas than may otherwise be beneficial for innovation. For instance, a designer might write a text prompt and hit generate. Upon viewing the result, they might adjust a few words in their prompt and hit generate again [20]—repeating this process akin to using a slot machine [2]. Their final prompt tends to be conceptually similar to their original one, often with small, incremental modifications aimed at getting the AI to create their initial design intention, instead of exploring new diverse design spaces [63]. As found in prior work, unsupported text-only prompting can be a limiting interface for generating good outputs from GenAI [70].

To understand how professional designers might experience these challenges with GenAI in their own processes, we conducted a comprehensive day-long exchange session with a team of professional designers from a large automotive company. The team outlined their design process, from conceptualizing a product design sketch based on a specific guiding theme to presenting final designs to stakeholders. From our discussions with the designers, we identified three key challenges in their use of GenAI:

1. Designing through text prompts feels unnatural as compared to their traditional sketching and ideation process.

2. T2I models struggle with generating inspiring designs from abstract concepts (e.g., "protective" car), a technique the team uses to create more novel designs.
3. It is difficult to directly build on generated designs—they appear "too complete", potentially leading to fixation.

To help designers working with GenAI avoid design fixation, we might take inspiration from design research working to break designer fixation. Two common strategies in line with what we learned from the professional design team include analogy-driven design [30, 36, 53], where concepts from outside the domain area are used to foster inspiration in the domain area (addressing C2) and providing multimodal interfaces that work visually and with lower-fidelity assets (addressing C1 & C3).

Several recent works in HCI have focused on developing new interfaces for GenAI to help designers explore a larger design space, including interactive prompting which aims help people who often have limited prompting ability [11], multimodal search [61], and visually navigating a 2D latent space interface [24]. Nonetheless, being able to help designers move beyond one thread of thinking and explore a wider design space while still taking inspiration and building on generated designs remains a significant limitation preventing GenAI to be fully integrated into designers' work. While the ML community offers methods to increase editing freedom of generated images, including local inpainting [57] and instruction-based image editing [12], these methods remove designers away from their natural rhythm of interaction with ideas.

In this paper, we introduce a workflow of continuous exploration with T2I models, encouraging designers to adopt a mindset that facilitates more iterative and exploratory design generation. Specifically, we built Inkspire, a proof-of-concept tool that provides a more familiar interaction built around iterative sketching and by leveraging the concept of analogical design to facilitate inspiration around design ideas. Our tool integrates analogical inspiration to promote concept-level ideation from abstract themes, allowing designers to recognize creative possibilities without needing to come up with them and write prompts manually. This reduces

cognitive friction, enabling fluid exploration of new ideas [14]. Additionally, inspired by prior works on drawing assistance systems that convert photographs into sketches to teach people how to draw [46], we introduce a new mechanism that converts high-fidelity AI designs into high-quality but low-resolution sketch scaffolds, directly underlaid on the designer's canvas. The scaffolding provides a transparent view into the current state of the AI and helps designers build on AI designs without being overly fixated on photorealistic renders. Finally, we enable a new design generation every time a new pen stroke is drawn (with near real-time performance), encouraging designers to consider small but meaningful changes in form and refinement with each pen stroke, thereby creating many more opportunities for designers to explore new directions. All of these components are designed to be seamlessly integrated into the sketch process familiar to designers.

To understand how designers use Inkspire, we invited both professional designers and novice users to design everyday products. We also asked them to use a baseline condition of a state-of-the-art ControlNet [72], which allows designers to use sketches and user-written prompts to guide generative design, but does not provide our proposed analogy inspirations or convert designs into lower resolution sketches, and requires the user to choose when they generate explicitly. The results show that users rated Inkspire as providing significantly more inspiration and exploration over baseline ControlNet. The interaction that users had with Inkspire was drastically different than using a baseline ControlNet—designers using Inkspire generated many more concept sketches and showed a process that appears more co-creative, whereas when using ControlNet they focused on manual sketching and refining prompts before handing off to the generative system. Inkspire enabled designers to more effectively co-create designs with T2I models with significantly increased partnership, controllability, communication, and sense of attribution over their creations. This research thus contributes:

- Inkspire, a proof-of-concept, iterative sketching tool that helps ground abstract concepts into analogies and converts AI-generated images into abstracted sketch scaffolds to support fluid design iteration and avoid fixation.

- A within-subjects study showing that Inkspire embodies a more iterative and exploratory workflow with T2I models, with designers rating Inkspire with significantly higher inspiration, exploration, and attributes of co-creation.

4.2.2 Related Work

This work builds on prior research in human-AI co-creativity for helping people generate ideas [52]. Specifically, we build upon prior works aiming to help designers create new visual concepts through text to image models and visual inputs [32, 51, 54]. We review how generative AI systems have been shown to increase idea generation, but counterintuitively can lead to more design fixation. We then review possible solutions for overcoming design fixation including analogy-driven design, computational sketching tools, and reducing the fidelity of generated images.

Generative AI and Design Fixation

One of the proposed benefits of generative AI is to help designers avoid design fixation, where designers remain stuck in a single line of thinking and a limited set of ideas, thus limiting the conceptual novelty of ideas and potential for innovation [35]. While many of the works suggest that generative AI can help people generate more ideas, potentially helping them to move into new conceptual spaces, recent experimental work has found contradictory results on generative AI's impact on design fixation. DesignAID [16] leverages large language models and image generators to help people explore a large, diverse space of ideas and was found to provide more inspiration than search-based tools. However, people rated the generative AI ideas as less valuable, with the paper authors suggesting that the ideas may not have been diverse enough, not well matched to the problem, or just not enough to break people design fixation. Bordas et al. [10] find that people using ChatGPT 3.5 to help generate ideas to protect and egg falling from 10 meters increased their idea generation but remained overly fixated on specific solutions. Wadinambiarachchi et al. [63] find that participants using a text-to-image generator to create a new chatbot avatar concept had significantly higher design fixation with fewer ideas, less variety, and less originality when using a text-to-image generator as compared to using an image

search engine or coming up with ideas unassisted. The authors suggest that people's limited prompting ability, a known issue when lay people try to use LLMs [70], led them to simply copy keywords from the design brief, limiting the language used to generate the images. Davis et al. [24] and Zhang et al. [71] report similar findings, where designers working with text-only image generators showed limited creative exploration, again due to people's limited abilities with text prompting.

To overcome the fixating issues of text-based generative systems, Davis et al. [24] developed a generative system taking example images of dresses and provided graphical buttons and sliders to generate more realistic vs. creative ideas and alter shape, color and texture. This visual generative system was preferred by designers and lead to more creative idea explorations. However, the high-fidelity images presented by many image generation tools, which Wadinambiarachchi et al. [63] suggest could lead to more fixation based on prior design research showing that high-fidelity images lead to more fixation over rougher sketches [18, 21]. In our work, we explore how to overcome the fixating issues that many generative AI systems have today. First, we look to help designers explore more novel ideas by providing interfaces that overcome their limited prompting abilities and help them generate more novel ideas. For this, we look to analogy-driven design as a potential solution. Second, we move away from text-only generation and provide more visual interfaces for design exploration (C1), helping better match how designers come up with ideas. We explore how iterative, computational sketching can be used as an visual input to image generation. Third, we break from showing only high-fidelity images which could lead to design fixation and explore how AI generated images can be altered and presented to designers in lower resolution forms to see if such representations may scaffold new ideas.

Analogy-Driven Design

Analogy-driven design, or design-by-analogy, is an approach to drawing inspiration from a known domain, including concepts and products, to find novel solutions to a target domain. An abundance of text-based research systems and prototypes such as DANE [31], Idea-Inspire 4.0

[60], and BioTRIZ [62] have been developed to support design ideation by offering different design-by-analogy capabilities that include retrieval and mapping of analogies to a target problem based on inferred similarities.

Recent proliferation of text and image data repositories combined with advances in vision and language models gave rise to new multimodal approaches that expand analogy-driven design to the visual domain [36]. For instance, Kwon et al. developed an approach that leverages visual similarity to discover visual analogies for generating new ideas [43]. In addition, Zhang and Jin proposed an unsupervised deep learning model, Sketch-pix2seq, to extract shape features from Quickdraw sketches, creating a latent space that enables defining visual similarities and searching for analogical sketches [74]. Jiang et al. developed a CNN-based model to create feature vectors representing patent images, which combine visual and technological information to enhance visual stimuli retrieval [37]. While these models present promising methods to support image-based analogy-driven design, the specialized nature of these models reduces their practical appeal to designers seeking exposure to out-of-distribution inspirations.

Large pre-trained models provide exciting opportunities to support domain-agnostic analogy based image retrieval. While T2I models alone struggle to generate images from an abstract concept such as "mystery" [40], machine learning research has demonstrated the use of LLMs to convert abstract concepts into semantically meaningful physical representations, thereby streamlining the process for downstream T2I generation tasks [28, 49, 66]. For instance, Fan et al. introduced an approach that extends an abstract concept such as peace with concrete objects (e.g., white doves, olive branches), then rewriting the original prompt to incorporate the objects in a scene [28]. Liao et al. proposed the Text-to-Image generation for Abstract Concepts (TIAC) framework that builds on a three-layer artwork theory to clarify the intent of the abstract concept with a detailed definition, then transforming it into semantically-related physical objects and concept-dependent form [49]. However, much of this work focused on scene illustration with multiple objects. Moreover, techniques that rely on automated prompt enrichment reduce the steerability of T2I models. In Inkspire, we apply a similar process through

the use of LLMs and and analogical reasoning to convert an abstract concept into individual physical objects from across multiple domains. We investigate how providing users with a menu of analogical concepts can help the user rapidly generate diverse analogical-grounded designs with T2I models, to overcome the current challenges of creating inspiring designs from abstract design prompts (C2) in ways that also offer them more control over design space exploration.

Guided Sketching

Research has explored various computational techniques that provide sketch guidance to aid in several use cases including skill building, serving as reference points, and encouraging creative exploration.

Many existing systems have explored guided sketching tools that help novices learn how to sketch. ShadowDraw [46] offers real-time shadow-based feedback, while systems such as The Drawing Assistant [34] and Painting with Bob [8] focus on translating photographs to sketches. Several works have explored crowdsourcing-based approaches. Systems like Limpaecher et al. [50] and Sketchy [58] leverage collective human knowledge to provide guidance. These systems excel at teaching specific drawing techniques but are less suited for open-ended creative exploration. Similarly, portrait-specific systems such as DualFace [33], which employs a two-stage drawing guidance for freehand portraits, and PortraitSketch [67], which provides face sketching assistance, demonstrate the value of domain-specific guidance but are constrained to a narrow use case.

Most relevant to Inkspire is Creative Sketching Partner [23], which retrieves sketches that are visually and conceptually similar to the user's sketches as a means to stimulate exploration and inspire new designs. This system demonstrates the potential of computational guided sketching systems to inspire new designs. However, it has the limitation of relying on existing sketch databases, whereas in this work, we leverage the generative capabilities of AI.

In our work, we draw inspiration from empirical research by Williford et al. [65], whose analysis of 240 concept sketches revealed that ambiguous sketch underlays could reduce fixation on conventional forms and promote divergent thinking, guided sketching techniques such as ShadowDraw [46], and existing design practices of sketch scaffolding [19]. With Inkspire, we propose a novel computational pipeline for converting GenAI designs into abstracted, yet high-quality sketching scaffolds that underlay the user's canvas. To the best of our knowledge, this work is the first to support the full closed-loop-cycle of sketching GenAI designs and abstracting GenAI designs into sketches. Through this technique, we address the challenge of building on GenAI images that are "too complete" (C3) by drawing designers' attention away from the high-fidelity generated image and inspire designers to iterate on top of the silhouette of GenAI designs with low friction.

4.2.3 Formative Study

To understand how professional designers use Text-to-Image (T2I) models in their work, we conducted a day-long exchange session with a team of seven professional product designers from a large automotive company. The designers in our exchange session work at a top 5 automotive manufacturing company and cover multiple disciplines of training spanning a wide range of roles in the company, including creative director, modeling lead, interior designer, exterior designer, artistic creator, conceptual lead creator, UX/UI and strategy. There is significant cross-collaboration across departments globally and with other companies in the industry. The designers also have experience in using T2I tools such as Midjourney [4] and Vizcom [5]. The designers showed our team their design processes in presentations with specific examples from their past work. This included sketches, concept boards, and various documentations of collaborative meetings for a wide range of mobility concepts. The designers also showed their process of using current T2I tools, and we discussed new ideas on how to design tools to support them. From our interaction with the designers, we identified key challenges they face when using T2I models and summarized them into three design goals to inform the development of Inkspire.

Design Goals

Design Goal 1. Sketching as a Natural Method of Interaction.

Designers emphasized that prompting is an unnatural approach to designing. They expressed difficulty in effectively conveying design ideas through language [70]. They often felt constrained by the need to craft comprehensive prompts, which limited their ability to explore a wider range of ideas. In contrast, designers expressed a preference for approaching design tasks through sketching [15], often beginning with just a simple line or silhouette. Therefore, our first design goal is to allow designers to interact with AI via sketching as a natural method of interaction. We aim to support designers in starting with simple abstract lines and assist them in progressing towards complete sketches.

Design Goal 2. Visually-Concrete Inspirations.

Designers mentioned that design briefs are typically inherently abstract, for example, "design a vehicle that conveys a sense of protectiveness". However, they found T2I models to generally produce poor and generic results when prompted with such abstract terms [69]. Even when resorting to prompt engineering tricks, they find it challenging to visualize abstract concepts in concrete forms. Therefore, our second design goal is to assist designers in visualizing abstract design themes through visually-concrete inspirations. Inspired by the way designers may draw inspiration from nature [26], we aim to recommend analogical inspirations [39] to designers and make it easy for them to quickly visualize a variety of diverse inspirations.

Design Goal 3. Complete the Feedback Loop.

Designers expressed difficulty in iterating on AI-generated designs and they described the process of using T2I models as a one-way process. They feel that the generated designs look "too complete", making it difficult for them to envision new ways to build on them. Often, designers find themselves in a position where they can either choose to use a design or discard it entirely. Therefore, we aim to bridge the gap in the feedback loop – while T2I models

transform ideas into images, our goal is to transform images back into abstractions (i.e., sketches), to allow designers to continue the iteration process.

4.2.4 System Implementation

We first illustrate how a user would use Inkspire through a concept car design example. We then describe the technical implementation of Inkspire, which consists of two primary components: Sketch2Design (generating AI designs from sketches and analogies) and Design2Sketch (converting AI designs into lower fidelity sketch scaffolding).

System Walkthrough

DeLorean is an automotive designer tasked with creating a concept car design that embodies a sense of "protectiveness" (Figure 1).

Ideating a Design Concept.

To begin, DeLorean uses the Analogy Panel (Figure 1 a) to generate visually-concrete inspirations for the abstract concept (Design Goal 2). He uses "car" as the subject and "protective" as the abstract concept, then clicks on the inspiration button. He is presented with a selection of inspirations, color-coded by categories: nature, fashion, and architecture. DeLorean experiments with several inspirations (such as bunker, armadillo, tortoise) by clicking on them. For each inspiration he selects, the AI generates a design in the Evolution Panel (Figure 1 c). He can manually edit inspirations in the concept box and manually click on the generate button. DeLorean decides to use tortoise as his inspiration.

Iterating on Designs through Sketching.

DeLorean now iteratively guides the AI through sketching in the Sketching Panel (Figure 1 b) (Design Goal 1). He starts off with a simple silhouette line, and the AI generates an initial design in the Evolution Panel. In the Sketching Panel, DeLorean sees a scaffolding abstracted from the initial AI design (Design Goal 3). This allows him to take inspiration from the AI designs without

being overly fixated on photorealistic renders. He is drawn to the bold curve of the windshield area shown in the scaffolding and loosely traces this part to add to his sketch. The AI then generates a new design and scaffolding.

DeLorean repeats this back-and-forth process with the AI (Design Goal 1), continuing to iterate on his designs through sketching until he achieves a design that satisfies him. To navigate between iterations, restart a sketch, or refine parts of the sketch, he uses the Undo, Clear, and Eraser tools. To explore design variations using the same input (the same inspiration and sketches), he uses the Remix tool.

Sketch2Design

The Sketch2Design component helps users brainstorm design concepts and generate product designs through sketching (Figure 2).

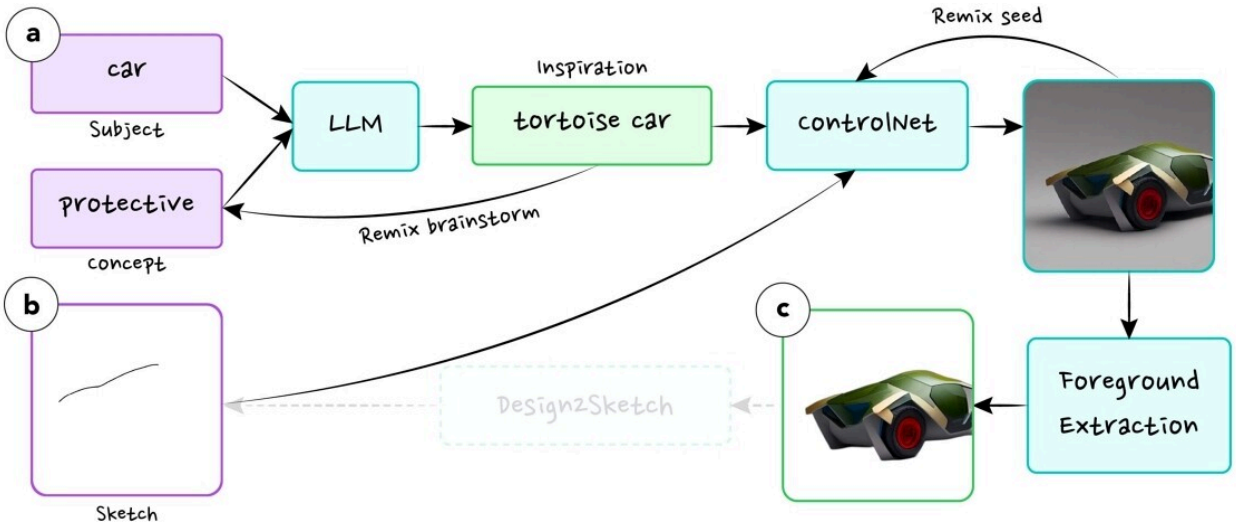

Figure X: Sketch2Design pipeline, including (a) inspiration generation with LLMs, (b) sketch-guided design generation, and (c) foreground extraction.

First, the user specifies the subject that they are designing for (e.g., car) and an initial abstract concept (e.g., protective). To brainstorm more concrete design ideas for the abstract concept (Design Goal 2), we leverage Large Language Models (LLMs) (GPT-4) [13] to generate analogical inspirations (Figure 2 a). We leverage prior techniques in chain-of-thought reasoning [64] to

break down the problem of creating analogies based on the given concept word. We take a two-step prompting approach shown in the listings below. We first prompt the LLM to detail the design principles for the given subject (e.g., car design).

> Describe the key design principles in <subject> design inone short paragraph

Such design principles can be useful as context [47] to ground the LLM to ideate inspirations that are more suitable for the specific product domain. An example intermediary result from this step for the domain of car design could be

> Key design principles for car design include aerodynamicsexteriors for fuel efficiency and performance...

Given the design principles, we then prompt the LLM to generate analogical inspirations. Our definition of analogy draws on the work of Gentner [29]. This definition involves identifying parallel relations from a source domain to apply to a target domain even when their surface features differ. We have structured our prompt using this definition of analogies by finding visually concrete objects from three source domains (nature, architecture, fashion) that convey concepts of the target domain (abstract concept).

> You are a <subject> designer. The design principles in <subject> design are as follows: <design principles from Step1>. Brainstorm analogical inspirations for <subject> designto convey a sense of <concept> from one of the followingdomains: nature, architecture, or fashion. Answer in abullet-point list of 10 items (item1\nitem2...\nitem3) usingvisually-concrete objects not adjectives and don't repeat.

We empirically found that prompting specifically for the domains of nature, architecture, and fashion leads to particularly interesting inspirations. Furthermore, these domains are outside of the primary product design domain and are common sources of inspiration for designers. The

LLM then provides single noun-phrases as results (e.g., protectiveness & → tortoise, armadillo, armor).

The user may select a recommended inspiration and continue branching out to explore further inspirations. For example, selecting tortoise and rerunning the analogy inspiration chain could result in new analogies such as tortoise → tank, backpack, treasure chest). The user may also freely change their concept (e.g., changing protectiveness to freedom, resulting in a new set of analogical inspirations). In our current implementation, users cannot return to previous inspirations or explore multiple inspiration branches in parallel. When a user selects an inspiration, it serves as a base to generate another set of inspirations. Furthermore, the generated inspirations remain independent of what the user sketches on the canvas. These features suggest potential areas for future work.

After selecting an analogical design inspiration, the user may create product designs by sketching on a canvas. Our conversations with the professional design team revealed that they often start a design with a single silhouette line (Design Goal 1). In Inkspire, the user may start generating images with as little as a single stroke (Figure 2 b). Using ControlNet [72] to guide Stable Diffusion [56], we generate a product design guided by the initial stroke. The user may continue adding additional strokes. Each time a stroke is drawn, we generate a new design, making the creation process iterative and implicitly encouraging users to focus on sketching instead of engineering text prompts (i.e., the current paradigm of working with T2I models).
ControlNet does not support per-stroke interaction out-of-the-box as it struggles with incomplete sketches, which is especially problematic during the initial stages of user sketching. Thus, we adapted the ControlNet model with a dynamic guidance scale to enable our desired per-stroke interaction (Equation 1). The guidance scale is a parameter for controlling how closely the model adheres to user input. We initialize with a low guidance scale to handle incomplete sketches. We progressively increase the guidance scale as the designer adds more ink to the sketch, pushing the generations to become more sensitive to the user's sketch over time.

$G(n) = 7 - 4 \cdot 0.5^{n3}$

where G(n) is the guidance scale at n number of strokes by the user. The guidance scale starts at 3 when n=0 and approaches a maximum of 7 as the sketch becomes more complete (n ≈ 10). The decay term creates a sharp logarithmic growth in guidance scale that converges gradually at the maximum value.

By gradually increasing the guidance scale, Inkspire allows the user to more precisely guide the generations as their sketches become more complete and well-defined. We maintain the same initial seed (seed specifies the random noise used to initialize image generations) between generations to maintain consistency between iterations and for near-real-time generations. The user may click on the "remix" button to change to a different seed and generate more diverse designs that break from the current thread that the designer is exploring. Finally, we remove unnecessary backgrounds from the generated design using a foreground extraction method [55] (Figure 2 c).

Design2Sketch

The Design2Sketch component helps users build on top of previously generated designs by converting them into scaffoldings by abstracting a design into a reduced-fidelity sketch-style with the aim of reducing design fixation on the high-fidelity image (Figure 4). The scaffolding appears as a transparent underlay beneath the user's canvas, functioning like tracing paper [3] that updates in real-time as they sketch. This enables the user to draw inspiration from aspects of the previously generated designs and also helps them overcome the challenge of starting with a blank canvas [38], especially during the early stages of sketching. The user can continue iterating through sketching, completing the sketch-to-design-to-sketch feedback loop (Design Goal 3).

While there are many methods for reducing high fidelity images into lower-resolution, we introduce a novel approach for converting designs to sketch scaffolds. We initially tested existing methods including Canny edge detection [17], HED soft edge extraction [68], a state-of-the-art method for extracting main lines from manga illustrations [48], and a neural network method explicitly trained on pairs of sketches and images. (see Figure 3). We observed that edge extraction methods including Canny edge detection and HED soft edge extraction frequently produce unwanted lines caused by the texture of designs. Furthermore, we observed that manga line extraction methods, trained primarily on cartoon illustrations, can lead to a loss of key lines or produce broken lines. Finally, neural networks explicitly trained on pairs of sketches and images on the task of translating images to sketches can create artifacts of excessive sketch stylization, such as shading effects.

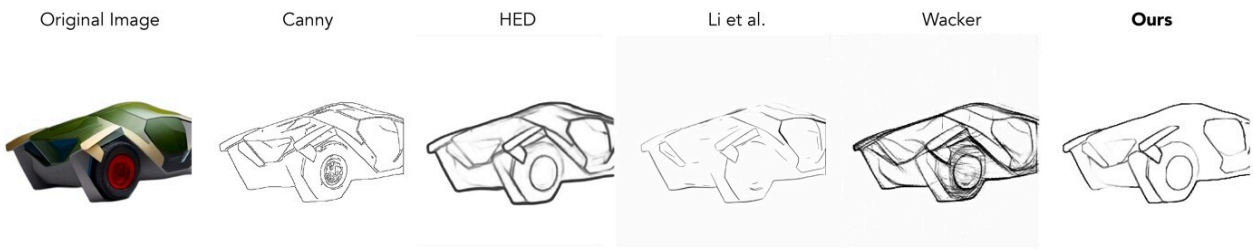

Figure X: Comparison of our Design2Sketch method with potential alternative methods, such as edge detection, manga line extraction models, and models trained explicitly on pairs of sketches and images.

In our approach, we combine semantic segmentation and edge extraction. First, we perform semantic segmentation on the design [41] to create a segmentation map that color-codes a design into distinct semantic regions (Figure 4 a). Given the segmentation map, we draw the boundaries between the different regions to create an image of semantic boundary lines. Second, we extract soft edges from the design [68] (Figure 4 b). These soft edges include varying thickness and line weight, simulating a sketch-like look, though often with many redundant lines caused by texture. Finally, we take a pixel-wise intersection between the segmentation map boundary lines from the first step and the extracted soft edges from the second step as the final scaffolding (Figure 4 c):

<equation>

where D is the generated design. Through this approach, we are able to acheve the best of both worlds – creating a sketch scaffolding that achieves a natural sketch look while focusing only on the design's key structural lines, filtered through the boundary lines from semantic segmentation step.

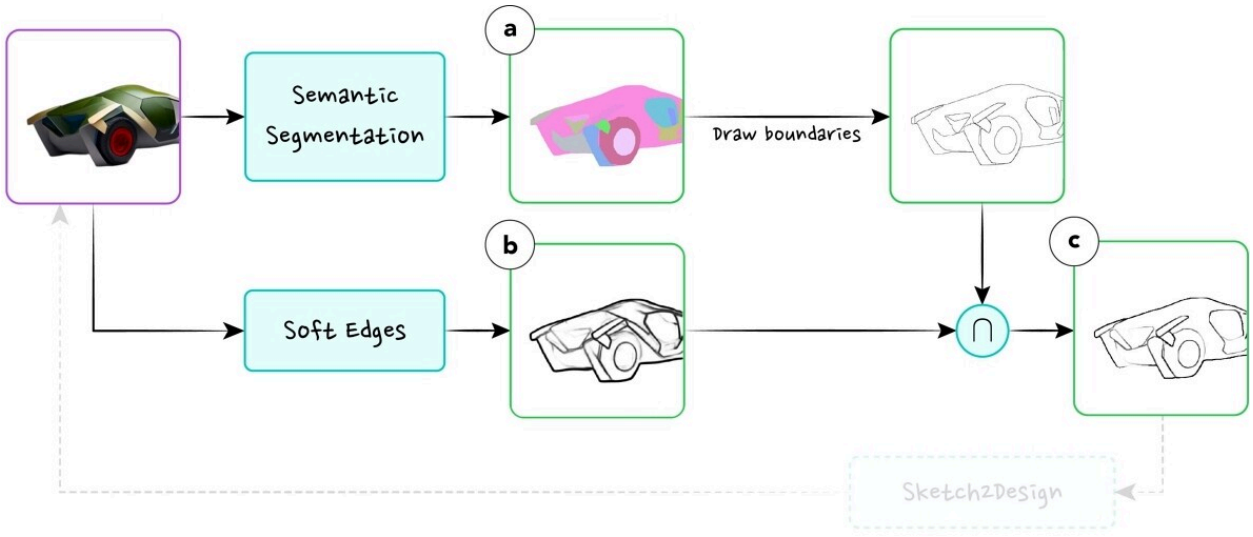

Figure X: Design2Sketch pipeline, including (a) semantic segmentation, (b) soft edge extraction, and (c) computing an intersection.

4.2.5 User Study

We conducted a within-subjects study to understand how Inkspire could address designers' pain points in working with T2I models, its potential to be integrated into design workflows, and identify areas for improvement. We compared Inkspire against a baseline condition using a typical ControlNet [72] setup consisting of a prompt box and a sketching canvas, adopting a similar interface layout as Inkspire.

Participants

We invited twelve participants (P1-P12, 10 male and 2 female) to participate in a one-hour user study. Among the participants, six are professional designers who perform product design

activities daily or weekly (self-rated confidence in product design μ =6.17, σ =0.75; self-rated confidence in drawing μ =6.00, σ =1.27; 7-point Likert scale) and six are novices who have moderate drawing experience (self-rated confidence in drawing μ =4.17, σ =1.33; 7-point Likert scale) but do not actively engage in product design. The participants were recruited through known contacts and Upwork [1], a platform for hiring freelancers. They were not exposed to the Inkspire system or concept prior to the user study. Participants accessed Inkspire and the baseline tool through a web browser.

Measures

For both conditions, we asked participants to complete questionnaires to capture their perspectives on using both Inkspire and ControlNet. We assess creativity using the Creativity Support Index [22], measuring exploration, inspiration, engagement, expressiveness, tool transparency, and effort/reward tradeoff. We assess designers sense of the human-AI collaboration using questions from [44] measuring controllability, communication, harmony, partnership, attribution, and ownership. We also asked participants to rate their experiences of sketching by referencing the sketching principles from Bill Buxton's Sketching User Experiences [15], measuring how quick and timely, inexpensive and disposable, and loose and abstract sketching with each tool felt. We asked designers to rate the quality of the final design created with Inkspire and ControlNet and their overall experience satisfaction using each tool. All questionnaire questions were rated on a 7-point Likert scale (7=highly agree, 1-highly disagree). We compared the questionnaire measures using parametric paired t-tests. In addition, we log user interaction data, such as when participants draw a new sketch stroke, edit a prompt, and generate a new design. (Figures 13).

Procedure

Introduction (5 minutes).

Participants provided informed consent and were given an overview of the study procedures. In addition, we briefly explain how components of the system, such as the analogy generation and sketching and scaffolding interaction, works.

Design Tasks (45 minutes).

Participants completed a design task with Inkspire (Figure 1) and another design task with the baseline ControlNet tool. The two design tasks are "design a lamp with the theme of serenity" and "design a chair with the theme of fluidity." We counterbalance both the order of the tools and the order of the design tasks. After each condition, the participants completed the questionnaires.

Post-Study (10 minutes).

Participants gave feedback during a short interview as well as through a free response questionnaire on the individual subcomponents of Inkspire, their overall experience of using Inkspire, whether they could see Inkspire being integrated into their design workflow, and areas for improving the tool. We reviewed these qualitative data to support the quantitative results.

4.2.6 Results

Creativity

Participants felt that Inkspire improved support for creativity across some attributes of the Creativity Support Index (CSI), shown in Figure 5. Notably, participants reported significantly higher exploration with Inkspire (μ =5.83, σ =1.27) as compared to the baseline (μ =3.83, σ =1.64), ($t(11)=3.94$, $p<0.01$, $r=0.77$, $ds=1.13$). Participants also reported significantly higher inspiration (μ =5.92, σ =1.24) as compared to the baseline μ =4.00, σ =1.41), ($t(11)=3.44$, $p<0.01$, $r=0.72$, $ds=0.99$).

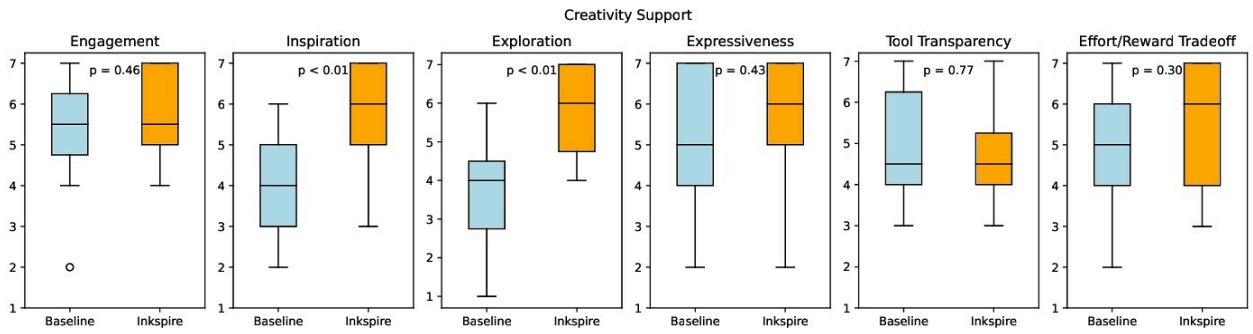

Figure X: Results on creativity measured with the Creativity Support Index (CSI) [22] (7-point Likert scale, higher is better).

From our interviews, participants noted that the [analogy] inspirations feature is "helpful while doing design ideations" (P12) and "a good tool to brainstorm in the early stage of design"(P5). Participants also found Inkspire to effectively support the exploration of multiple ideas, such as a "variety of forms, styles, patterns, and proportions"(P5). P11 explained that they could explore different design directions by manipulating high-level concepts ("I could just make a basic shape, and change a few keywords and the entire look and feel would change and present me with some great concepts"). In contrast, we observed that in the baseline condition, many participants focused on sketching extensively on a single idea, a potential sign of the "sunk-cost effect" [7] (the more time spent in a given direction, the harder it is to move to a different one).

We also note that the other attributes of the CSI, Engagement, Expressiveness, Tool Transparency, and Effort/Reward Tradeoff, were not significantly different between Inkspire and the ControlNet baseline. While Inkspire did not improve these aspects of creativity over using ControlNet it did not appear to degrade them either. Overall, we find that Inkspire improved exploration and inspiration, supporting our original design goals and working toward reducing design fixation.

Human-AI Collaboration

Designer self-ratings of Human-AI Collaboration.

Participants felt that Inkspire improved the experience of collaborating with the AI, across dimensions of Human-Machine Collaboration questions from [44] rated on 7-point Likert scales, shown in (Figure 7). Participants reported significantly higher controllability when using Inkspire (μ =5.58, σ =1.00) as compared to the baseline (μ =4.17, σ =1.19), ($t(11)=3.56$, $p<0.01$, $r=0.73$, $ds=1.03$). Participants felt that they had significantly higher communication with Inkspire (μ =5.75, σ =1.14) as compared to the baseline (μ =3.92, σ =1.38), ($t(11)=4.52$, $p<0.01$, $r=0.81$, $ds=1.31$). In addition, participants rated having a significantly higher degree of partnership with the AI when using Inkspire (μ =5.83, σ =1.03) as compared to the ControlNet baseline (μ =3.42, σ =1.56), ($t(11)=4.57$, $p<0.01$, $r=0.81$, $ds=1.32$). Lastly, participants rated having significantly more of their own attribution in the designs when using Inkspire (μ =5.25, σ =1.29) as compared to the baseline (μ =3.67, σ =1.07),($t(11)=3.51$, $p<0.01$, $r=0.73$, $ds=1.01$). Participants did not

report significant differences in their feelings of harmony with the AI or ownership over the designs between Inkspire and baseline ControlNet.

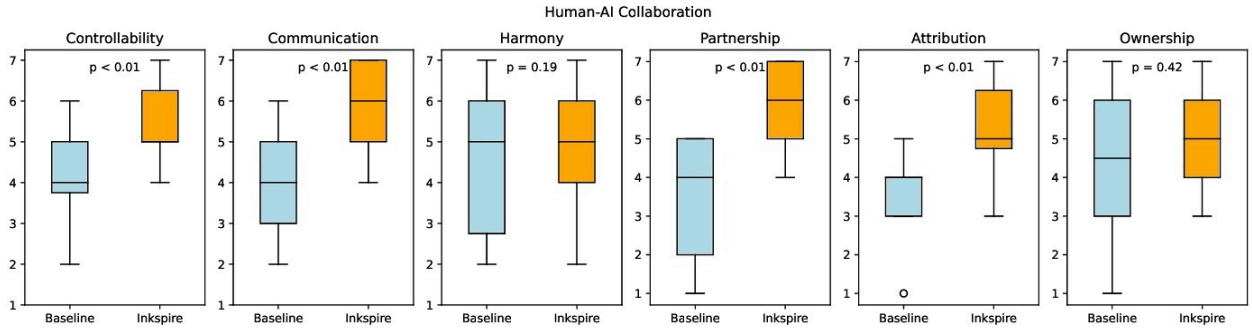

Figure X: Results on human-AI collaboration measured with Human-Machine Collaboration Questions from [44] (7-point Likert scale, higher is better).

From our interviews and open-response questions, many participants commented that they found the scaffolding helpful in being able to understand the current state of the AI and plan subsequent sketches ("[The scaffolding] was useful because it let me know where the current iteration was so I could tell where I'd like to move it next" (P11)). With scaffolding, participants felt that they could steer the direction of the design by building on previous generations ("[scaffolding] helped with building upon the previous AI-generated design and gave me a direction for what to adjust" (P6), "the AI-generated drawing overlay help[ed] me to draw my next line"(P3), "having the [scaffolding] as reference helped me [to] combine and remix [old designs] in my [new] sketches "(P6)). These observations and self-reported results suggest that Inkspire improves human-AI collaboration by increasing controllability, communication, and partnership.

Prompting Behavior.

As expected, we observed that participants generally did less prompt engineering when using Inkspire (μ =8.50, σ =6.31, number of prompts) than when using the baseline (μ =11.9, σ =10.6, number of prompts). When using the baseline ControlNet system, participants often relied on manipulating the prompt to change their design whereas they used more diverse analogical inspirations as prompts when using Inkspire. Analyzing the semantic similarity between user

prompts, we observed a much lower semantic similarity in the Inkspire condition (μ =0.51, σ =0.08) compared to the baseline (μ =0.76, σ =0.13), measured with BERTScore [73].

Looking at the logs of prompts designers used, we observed that participants using ControlNet often stuck to their original prompt, making small edits to make the prompt more and more detailed (see Figure 8). For example, P3 started with the prompt serenity lamp (i.e., explicitly prompting the AI with the abstract design task), and then expanded it with additions like serenity lamp with clear glass and black base, serenity lamp with clear glass and black pedestal, and so on. This result echoes the prompt fixation results of [63] and adds further evidence to the challenges designer have in knowing how to prompt [70]. Using Inkspire, participants made fewer manual prompt edits and frequently utilized the recommended analogical inspirations to guide their prompting direction. For example, the same participant above (P3) used various analogical inspirations for creating a fluid chair such as silk, river, and waterfall.

Sketching Behavior.

We observed that participants drew fewer total sketch strokes using Inkspire (μ =17.3, σ =7.40, number of strokes) as compared to using the baseline (μ =59.8, σ =40.5, number of strokes). When using Inkspire, participants also had a lower sketching frequency (μ =12.5, σ =12.5, strokes/min) as compared to the baseline (μ =20.1, σ =12.6, strokes/min), though this was not significant (t(11)=-1.67, p=0.12, r=0.45, ds=0.48). Participants also spent more time between strokes when using Inkspire (μ =13.8s, σ =12.1s) as compared to the baseline (μ =4.44s, σ =2.94s), (t(11)=2.66, p=0.02, r=0.63, ds=0.77).

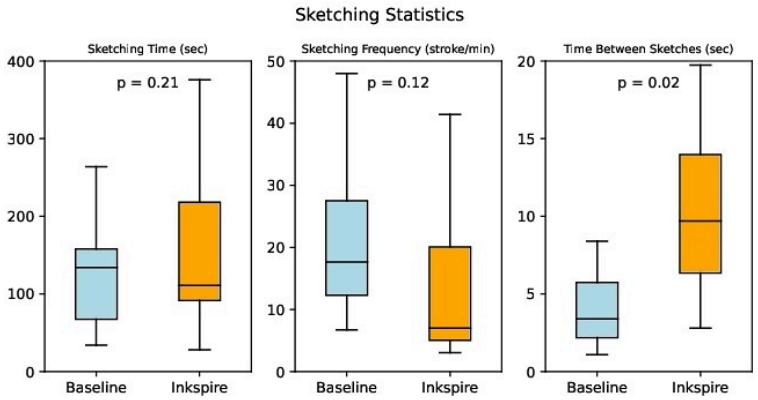

Figure X: Results on sketching statistics, including total sketching time, sketching frequency, and time between sketches.

Despite participants sketching less and taking longer between adding more ink to their drawing, participants rated that sketch strokes felt significantly more inexpensive when using Inkspire (μ =6.08, σ =0.90, 7-point Likert scale, higher is better, based on Buxton's sketching principles [15]) as compared to using the baseline (μ =3.75, σ =1.60, 7-point Likert scale, higher is better), ($t(11)=5.01$, $p<0.01$, $r=0.83$, $ds=1.45$). Participants also rated that the sketching was more abstract when using Inkspire (μ =5.75, σ =1.14, 7-point Likert scale, higher is better, based on Buxton's sketching principles [15]) as compared to the baseline (μ =3.83, σ =1.47, 7-point Likert scale, higher is better), ($t(11)=4.24$, $p<0.01$, $r=0.79$, $ds=1.23$). Overall, Inkspire appears to have provided designers with an interface for sketching ideas collaboratively with AI that improves over the more stilted sketching and image generation experience of other text-to-image systems.

Analogy Inspirations.

We observed that participants generally explored multiple analogical inspirations (μ =4.58, σ =2.43) distributed across the three categories (Nature μ =2.08, σ =1.56; Architecture μ =1.58, σ =1.3; Fashion μ =0.917, σ =0.716) (Table 1). Architecture was the most common final choice (n=6 instances across participants), followed by Nature (n=4) and Fashion (n=2), though participants explored Nature-based inspirations more frequently on average. We also observed that participants frequently switched between analogy categories. For instance, P12 quickly switched between all three categories. Among the switches, we observed that Nature→Architecture (n=6) and Nature→Fashion (n=5) were common transitions. This may suggest that Nature can serve as a common "bridge" category. Many participants' final category deviated from their initial category after exploration. For instance, P2 started with Nature and selected Architecture after exploring 14 different inspirations across all categories. Overall, participants drew inspiration from all categories, with Nature playing a central role, though each category proved to be a valuable source for inspiration.

Overall Usage Patterns.

By analyzing our logged interaction data, we observed common usage patterns for Inkspire vs. the baseline ControlNet workflow (please see Figure 10 for an illustrative example). First, participants using Inkspire often start the initial ideation phase by experimenting with many analogical inspirations to explore the design space (shown as many thought bubbles during the beginning, Figure 10). This early divergent exploration behavior may suggest that designers were able to consider a wide range of inspirational ideas at the start of their process. Subsequently, participants using Inkspire engaged in a highly iterative manner of sketching and interacted with scaffolding, often drawing one or a small number of strokes, seeing a generation and scaffolding, then pausing a bit to consider where to move next (shown as many pencils interleaved with sparkles, Figure 10). In contrast, when participants used the baseline system, they often did a complete full sketch (shown as large stretches of a single pencil, Figure 10) before ever hitting the generate button. Designers made more "Undo" actions, removing strokes that they had made (as shown in orange blocks). Overall, participants generally created more new designs with Inkspire than with the baseline condition (shown as significantly more sparkles in Inkspire than the baseline, Figure 10). Together, these patterns show how when participants used ControlNet, they focused more on crafting a whole sketch then handing it off to the generative AI rather than working collaboratively back and forth with the AI.

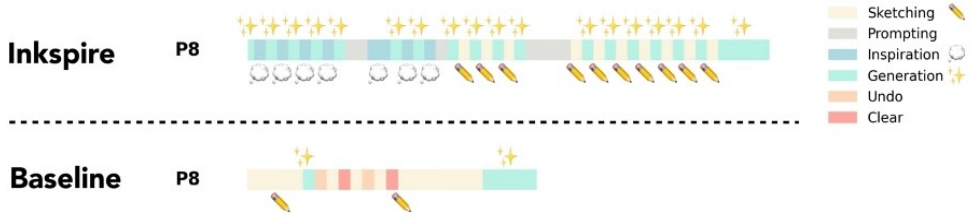

Figure X: Example user interaction log. We observe that, using Inkspire, the user started by ideating several analogical inspirations (thought bubble). Next, they performed sketching in a highly iterative manner (pencil+sparkles). Overall, the user generated a significant amount of new designs (sparkles). In contrast, using the baseline condition, the user sketched in large stretches with infrequent new design generations. We see that this pattern generally holds true across participants.

Final Design Quality

Participants rated their final creations as having higher design quality when using Inkspire (μ =5.92, σ =1.08) as compared to the baseline (μ =4.58, σ =1.93) on a 7-point Likert scale, shown in Figure 11 (left). Though this difference of ratings for design quality are not statistically significant, ratings showed less spread than when using the baseline (t(11)=2.11, p=0.06, r=0.54, ds=0.61).

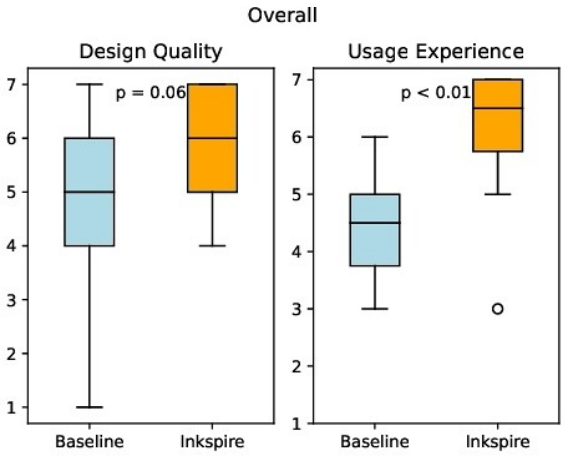

Figure X: Results on design quality (left) and usage experience (right) (7-point Likert scale, higher is better).

In Figure 6, we show several example creations from participants.

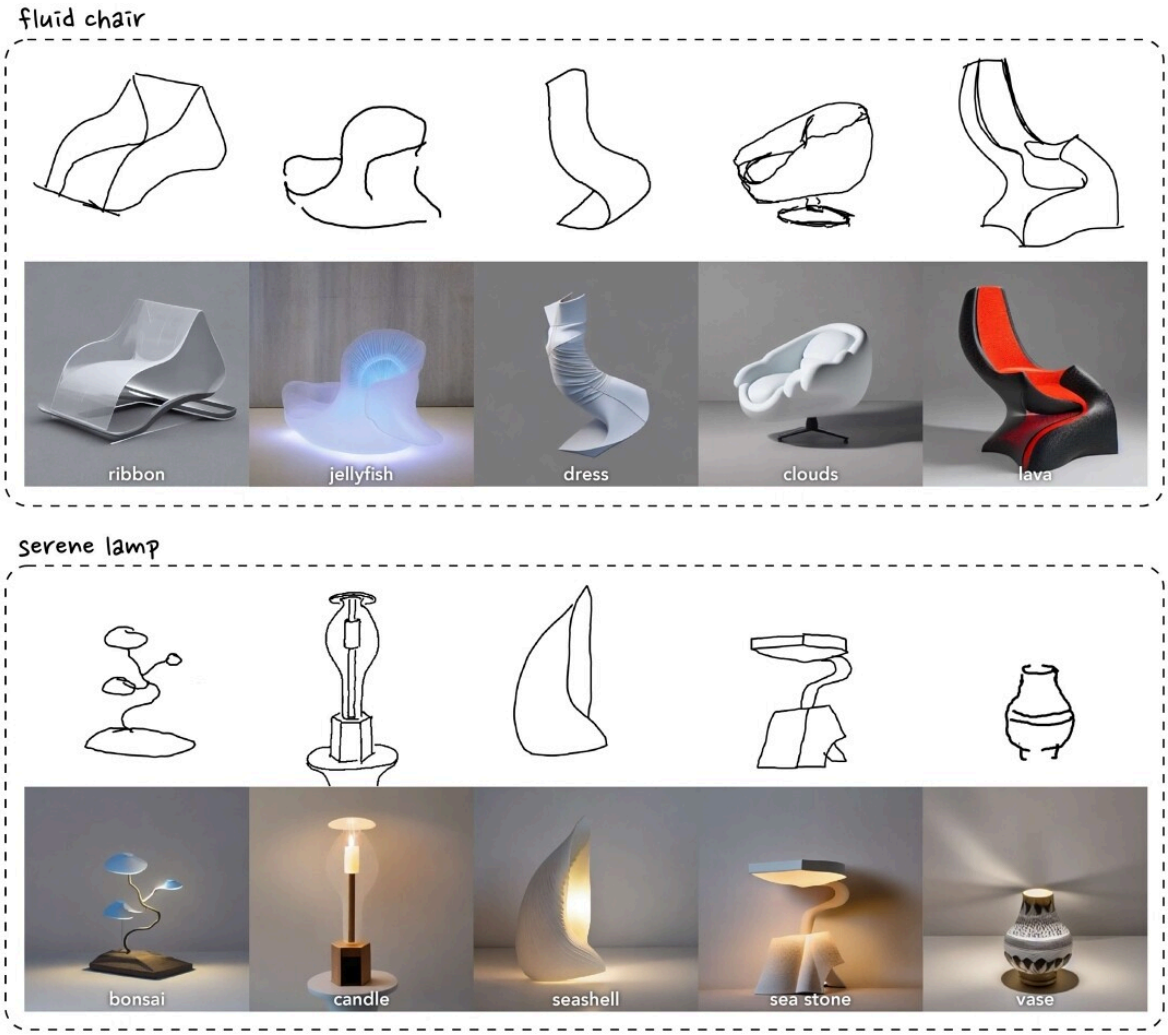

Figure X: Example designs created by participants using Inkspire for the design tasks of designing a fluid chair and a serene lamp. The final participant-generated sketch is shown on the top, and the generated T2I image is shown on the bottom, along with the selected analogy word chosen by the participant.

While the quality of output in general appears relatively high, we do observe that the final results produced with Inkspire do appear to be more diverse than the results produced with ControlNet. Looking at the trend of designs created during a user session, we see that the thread of designs created using ControlNet often do not change tracks, whereas those done with Inkspire show more diversity of conceptual exploration. An example showing chair designs from two different users illustrates this in Figure 8. The participant using Inkspire (top) shows a wide exploration of concepts for a transparent chair including glass, silicon, and jellyfish. Alternatively, a participant using ControlNet creates a modern Italian chair with a high quality

initial sketch and iterations on the prompt from lancia style to lancia style, iconic to lancia style, iconic forms. While both final result are generally of high quality, the final output from the participant using Inkspire is derived from a more exploratory process over design analogies over the more fixated design process from the designer working with ControlNet.

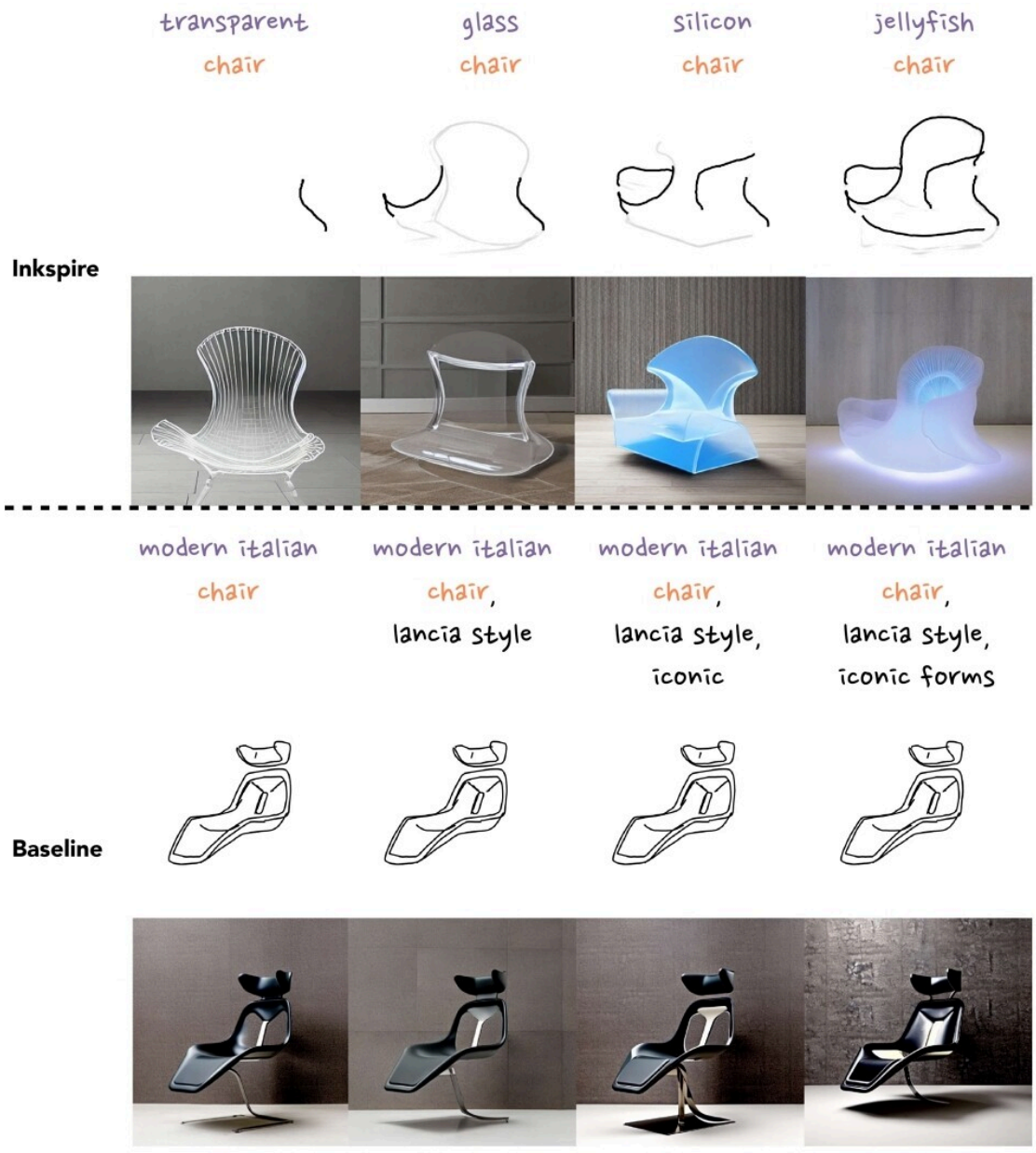

Figure X: Example user iterations with Inkspire (top section) vs. the baseline condition (bottom section). For each section, the top row shows the prompts, the middle row shows the sketches and scaffolds, and the bottom row shows the generated designs. Using Inkspire, users create diverse designs via analogies. In addition, users start with

a single sketch line and continuously build on their sketch, with the guidance of scaffolding. In contrast, using the baseline, users typically draw a full sketch and update their prompt with incremental modifications. This may lead to a higher degree of fixation and a smaller design space explored.

Designer Ratings of Usage Experience Satisfaction

Participants rated significantly higher satisfaction with their usage experience when using Inkspire (μ =6.08, σ =1.24) as compared to the baseline (μ =4.33, σ =0.98) when rated on a 7-point Likert scale ($t(11)=3.54$, $p<0.01$, $r=0.73$, $ds=1.02$), shown in Figure 11 (right).

Based on the interaction logs, Inkspire may improve participants' experiences of sketching with AI by supporting low cost experimentation and helping participants draw abstract sketches that focus on the big picture rather than being bogged down by small details. In the baseline, we observed that participants drew a number of large strokes in succession before generating anything, with the majority being "filler" lines. This generally led to designers spending more time between trial-and-error attempts to achieve their intended result, possibly reducing their satisfaction when using ControlNet.

4.2.7 Discussion

We introduced Inkspire, a prototype system to explore new ways for designers to leverage generative AI while avoiding design fixation and embedded into their existing practice of sketching new concepts. We focused on a challenge identified in design research literature that exploring a wider range of ideas can lead to improved design outcomes, though designers often find it difficult to naturally engage in diverse design exploration [27, 42]. Inkspire helped designers explore a more diverse design space by recommending them with diverse analogical anchors based on their initial design concept (e.g., turtle, shield, or bunker for embodying the concept of "protective"). These analogical concepts were used as input along with the sketch from the designer to generate new images. The designer could then continue to iterate on the designs collaboratively with the AI by leveraging stroke-by-stroke generation and building on low-resolution sketch scaffolds of high fidelity renders. Through a user study, we found that

participants using Inkspire explored a wider design space and qualitatively changed their ideation process to be more conceptually iterative and collaborative with the AI. In contrast, participants in a control condition, in which they used a state-of-the-art ControlNet, primarily focused their efforts on crafting full sketches and making small changes to prompt inputs before having the AI render an image.

A core challenge we grappled with in this work was helping designers to leverage generative AI while avoiding fixation on the outputs of the AI. We believed this to be especially likely in the context of creative design and sketching because of several factors derived from our formative conversations with professional designers and prior research: text prompting leading to what designers called "unnatural" interaction, text-to-image models struggling with generating inspiring designs from abstract concepts, both often due to designer's limited abilities to prompt well [24, 63, 70, 71] and the AI generating outputs that often look overly "finished", a well-known cause of design fixation [18, 21, 63].

Our results suggest a fundamental issue with the structure of prompt- or sketch-focused generative image interfaces, which is that current controls through prompts or sketches can engender an affordance of needing to feel relatively complete in order to convey the designer's intention. The results from when our participants used ControlNet are similar to results from Wadinambiarachchi et al. [63] where designers focus most of their effort on minor iteration of prompts. Needing to provide a rich enough prompt or a full enough sketch may explain why our participants felt less control over the ideation and less communication and partnership with the AI model.

In contrast, with Inkspire the design problem prompt and analogical keyword concepts provide a scaffolding in the design space allowing a single stroke to have expressive meaning while also generating an output that is relevant to the user's goals. Relatedly, our work also explores ways for designers to have a shared mental model of what they are trying to accomplish versus what the AI is trying to do. For example, an interesting point we noticed from our participants was

that they found the rough sketches inferred from the AI generation to be useful as control points that showed what the AI was doing and how the user could emphasize or change that, for example by either drawing over or changing a stroke on the roughed sketch, respectively (please see Figure 12). In this way the rough sketch serves not only as a scaffold for the designer's iterative sketching but as a communication tool and shared mental model with a generative AI partner. The continual updating of both sketch and analogical concept ultimately leads to a more iterative and collaborative process where the designer works in turns with the AI. Other generative AI tools could potentially build on this pattern of rapid turning taking to help promote more co-creativity for their creative task.

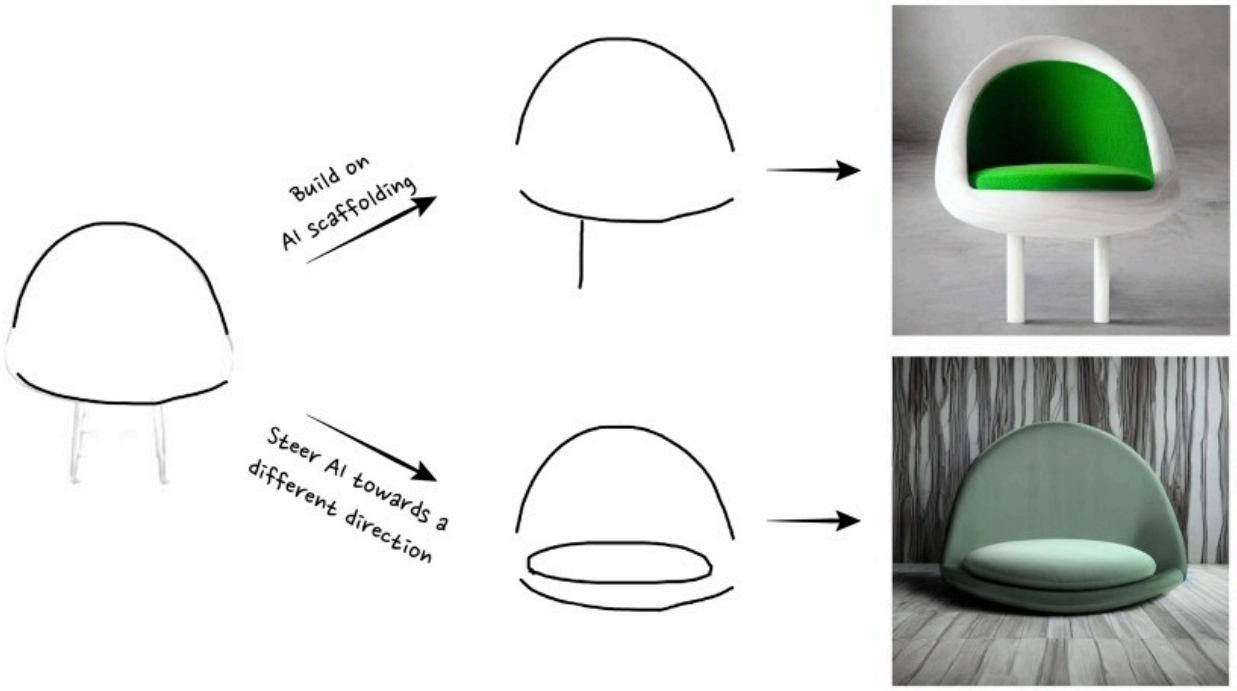

Figure X: Given a scaffold, the user may choose to build on it (by tracing it) or steer the AI towards a different direction.

Overall, the interactions we designed into Inkspire appear to have worked in helping our participants avoid design fixation. As commented on by P6 "the AI was helpful in providing alternative designs I did not consider previously, especially by adopting a new semantic understanding of the sketch that differed from my initial intention." However, even though

working with Inkspire pushed the designers into a space they had not originally considered and even resulted in them sketching less in total, they found that we also succeeded in overcoming issues where designers feel less agency when working with generative AI. That the participants felt they had more agency and more ownership of the final design outputs suggests the analogies, stroke-based interaction pattern, and scaffolds led to more fluid interaction through a design space. Again quoting from P6, Inkspire had "an interesting workflow that I think helps make the tool disappear more in comparison to a prompt-based drawing tool. It allowed me to focus more on sketching than prompt engineering."

While our overall results show that Inkspire achieved our design goals, there are some potential issues with Inkspire that could arise and could be considered in future co-creative systems. First, although we see more evidence of designers moving to more distinct ideas throughout a session, they still work along a single thread. Prior research has shown that designers often prototype ideas in parallel and that such parallel exploration can lead to better design outcomes [27]. One potential way to enable parallel explorations in Inkspire is to generate numerous analogies for a concept and generate an image for each during each sketch step. Second, as Inkspire was primarily designed for early explorations, it may not serve a designer as well during later stages of the design when they have a clear idea in mind that they are trying to render. In this case, a designer may prefer using a tool such as ControlNet where they can focus on the details they want before handing it to an AI. Lee et al. similarly proposed an adaptive multimodal T2I system that initially supports early ideation with a prompt-guided (e.g., protective chair) and sketch-supported (e.g., simple sketch) system, which gradually evolves into sketch-guided (e.g., detailed sketch) and prompt-supported (e.g., "more curved back") system to help refine the concept in later stages of the creative process [45]. This suggests potential opportunities for generative systems to provide controls for how much turn taking the AI should aim to have with the designer. Third, Inkspire and other generative AI approaches may help individual designers break their fixation on specific ideas, however, if common generative models are used, it may promote collective design fixation, where everyone’s images begin to look the same as they have a common source [6]. This being said, Inkspire may work to

counteract such collective fixation by promoting increased designer engagement by focusing more on suggestions of designs though lower-resolution sketch underlays rather than pixel-level in-painting. Future systems could explore other techniques for avoiding design fixation, such as showing partial photographs [21] instead of sketch underlays. Other ideas for reducing fidelity such as blurring or filtering could also promote designers to fill in details on their own, potentially leading to designs less similar to common output from AI models.

4.2.8 Limitations and Future Work

There are several avenues for future work on improving Inkspire collected from participants' comments. First, we could improve the sketching canvas with more advanced sketching features such as line weight control and shading tools, which could allow users to sketch with greater detail. Second, participants suggested adding more fine-grained region-based control to the Sketch2Design pipeline. For example, a user could specify different material types for different regions of the sketch. Third, we could extend the analogical panel to include the capability to go back to a previous inspiration and perform multiple branches of inspiration explorations in parallel. This could help users compare vastly different design directions. Fourth, we currently adopt a two-step prompting mechanism for generating analogical inspirations: defining the design principles for the target domain, then generating visually-concrete object inspirations for the target domain using nature, architecture, and fashion as source domains. Future work could explore more complex prompting structures, such as traversing hierarchical tree structures [39], that might lead to better analogical inspirations. Fifth, Inkspire contains several features that could all affect a user's behavior, such as automatic per-stroke generation and sketch scaffolding. For future work, we could conduct additional ablation studies. For example, adding dynamic guidance scale to enable automatic per-stroke generation for the baseline condition or hiding sketch scaffolding for Inkspire. Sixth, we adopted a within-subjects design with a relatively small sample size (n=12), which could be improved through larger-scale studies. Seventh, participants suggested the possibility of adding additional guidance for the AI, such as mechanical and material constraints, though such a feature may require more research into how to better connect analogically created T2I design concepts with their feasibility to be

manufactured. Finally, in this paper, we introduce a sketch-generation-scaffold interaction with generated analogies for abstract concepts through the application of product sketching. We think the sketching strategy could generalize to other forms of drawing, though the analogies might be better suited for product design. For example, very recent work has explored a similar strategy but using an image underlay instead of sketch scaffolding [59]. Future work could explore extending the system to additional design domains and incorporating background elements into designs for mockup or decoration.

4.2.9 Summary

Inkspire blends *iterative sketching* with *structural conditioning* via ControlNet and conversion of AI-generated designs into sketch scaffolds, creating a fluid back-and-forth between designer sketching and AI generation. Inkspire primarily improves the *responsive* characteristic of AI controllability (slow → real-time) and shifts upwards in *controllability* in the intuitiveness-controllability spectrum. In the next section, I outline my proposed work, which aims to increase granularity from whole-image inspiration to fine-grained blending of multiple atomic inspirations.

V. Proposed Work: Evolution as an Interface

5.1 Introduction

Consider a designer who wants to design a Japanese-style chair. They collect a moodboard of inspirations, trying to capture certain styles they want to see in their design by collecting images that evoke them.

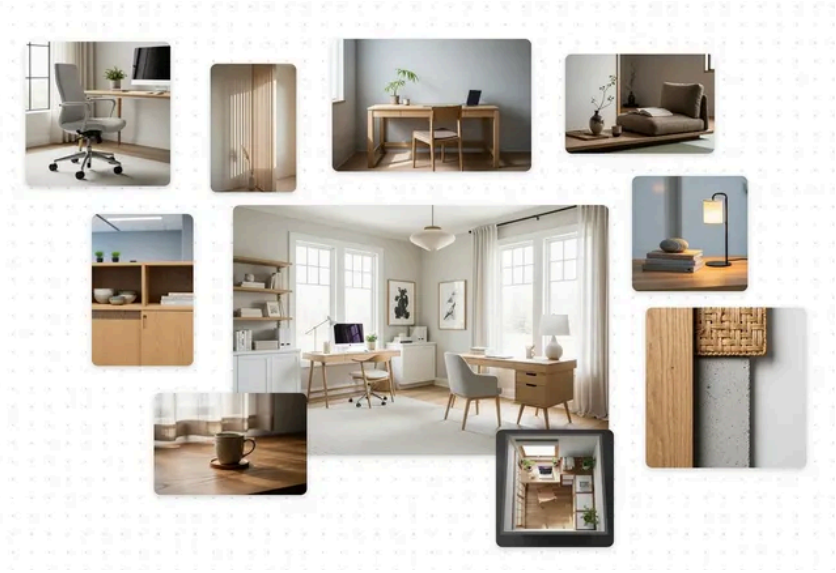

Figure X: Example moodboard of Japanese-inspired elements

However, when they collect inspirational images, they might only like a certain aspect of the images. For example, the texture of the bamboo wood, the coffee color scheme, or a somewhat zen aesthetic.

Figure X: Specific aspects of the moodboard images that inspire the designer

Current commercial tools only support full-object style transfer, such as techniques in image-to-image [Image2Image] and my previous work Inkspire [Inkspire], which loses a lot of granularity that designers actually desire. What designers might currently do is to manually try to crop it out, like cropping out the wood texture from the image, before doing AI style transfer, though this might be hard for concepts like color or abstract aesthetic. In my proposed work, I ask: How can I support style space exploration at a smaller granularity?

I propose StyleGenome, an AI-powered petri dish for mixing design styles at fine granularity. The key idea of StyleGenome is to encode design styles, like minimalism or luxury, as "genes" that can be mixed, mutated, and evolved through genetic operations. This provides a powerful new interaction for iteratively exploring style blends and discovering unexpected composites. My proposed work continues my research methodology of mapping interface paradigms to controllable mechanisms in AI models. Specifically, mapping a genetic evolution interface metaphor with parameter-efficient adaptation via low-rank adaptation (LoRA) and the ability to blend multiple LoRA-trained style models with finegrained weightings.

5.2 System Design

I have built an early implementation of StyleGenome and Figure X shows the interface. Here is an early demo: https://drive.google.com/file/d/154jD4Y9O-wy-xma2564xJUb3ACGowkV7/view?usp=sharing.

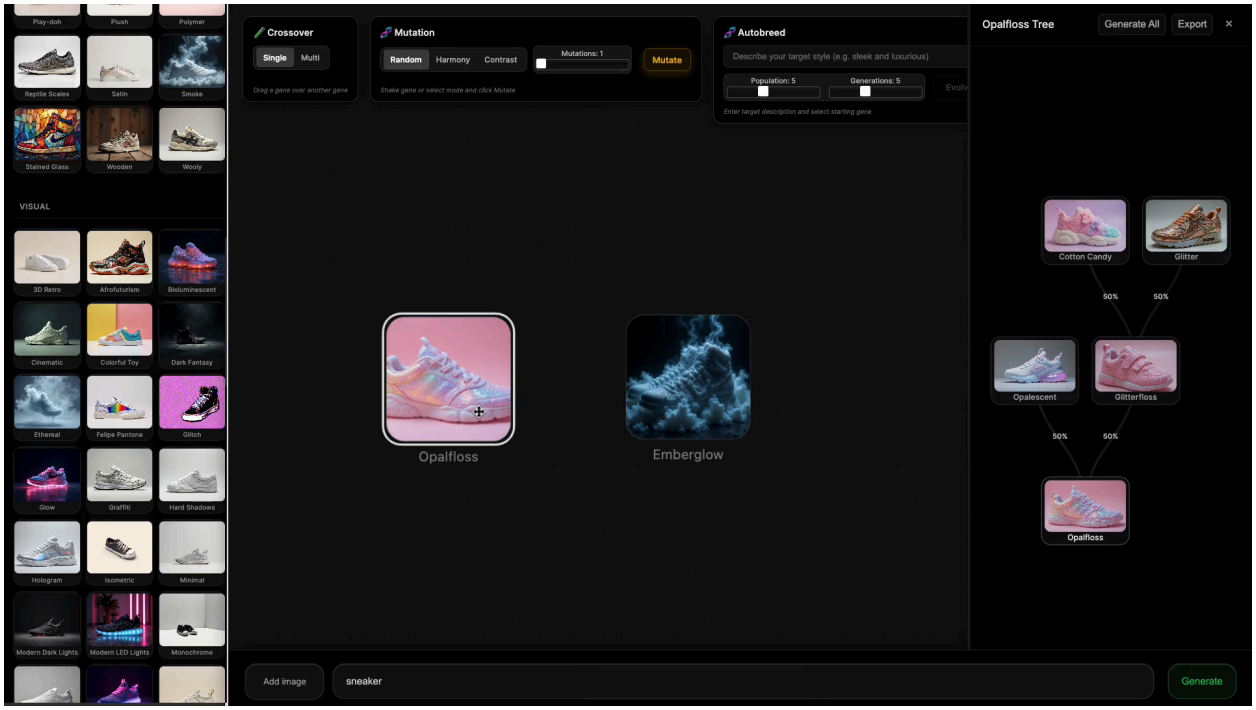

Figure X: StyleGenome Interface. The gene library on the left contains over a hundred style genes (FLUX LoRAs). The canvas on the right allows designers to mix genes together.

The StyleGenome interface consists of two main components: the gene library on the left and the canvas on the right. The gene library contains hundreds of different style genes curated from various domains, including materials and textures like bamboo and glass, visual styles like minimal and isometric, architecture genes like Zaha Hadid, and fashion genes like streetwear. These style genes are pretrained LoRAs and are ready to use. Designers can also bring in their own moodboards to easily train their own style genes, adding to this ever-expanding library. The canvas on the right is where designers can perform genetic operations on style genes, including crossover, mutation, and evolution.

Crossover

Crossover allows designers to mix two or more style genes together. For example, a crossover between cotton candy, fluffy, and colorful toy genes creates aesthetic blends.

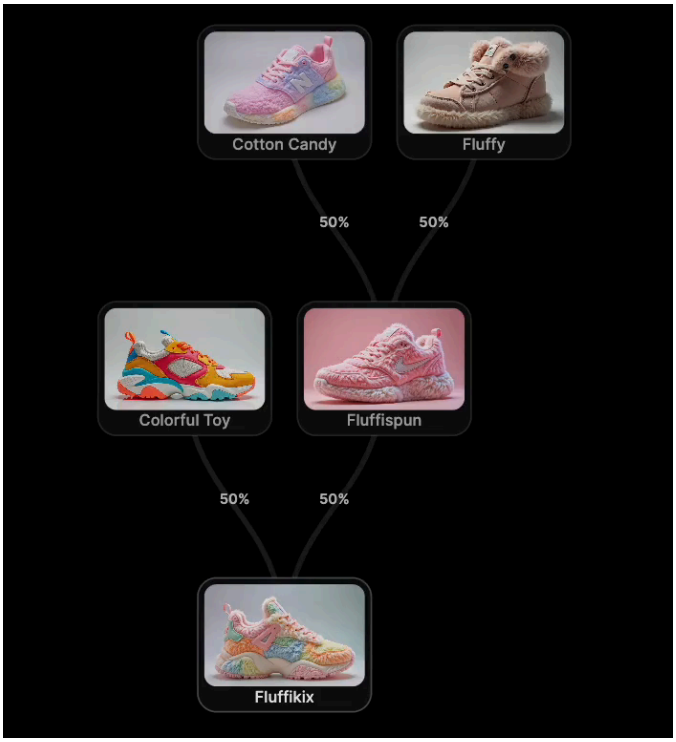

Figure X: Crossover of Cotton Candy x Fluffy x Colorful Toy genes.

Mutation

Mutation enables designers to see different variations of a style gene by using the initial style gene as a seed and pairing it with several random genes. For instance, mutations on the "Felipe Pantone" artist gene produce several variant interpretations.

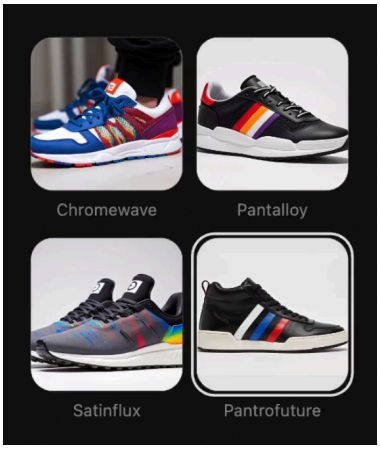

Figure X: Random mutations of Felipe Pantone gene.

Evolution

Evolution allows designers to perform automatic evolution toward a goal defined using natural language, letting StyleGenome automatically blend genes, apply natural selection, and evolve toward specified criteria. Evolution uses a genetic algorithm with configurable population and generation settings. With a population of three images per generation and three iterations of evolution, the system first selects three random genes and generates images, then compares each image with the user's goal using CLIP embedding similarity. The top candidates from each generation are selected to continue evolution, with each top candidate doing a crossover with another random gene to maintain the population size. This continues until the final generation, where the top candidate becomes the winner. For example, automatic evolution can design a sneaker that conveys the concept of "quiet luxury" through natural selection across several generations of crossovers.

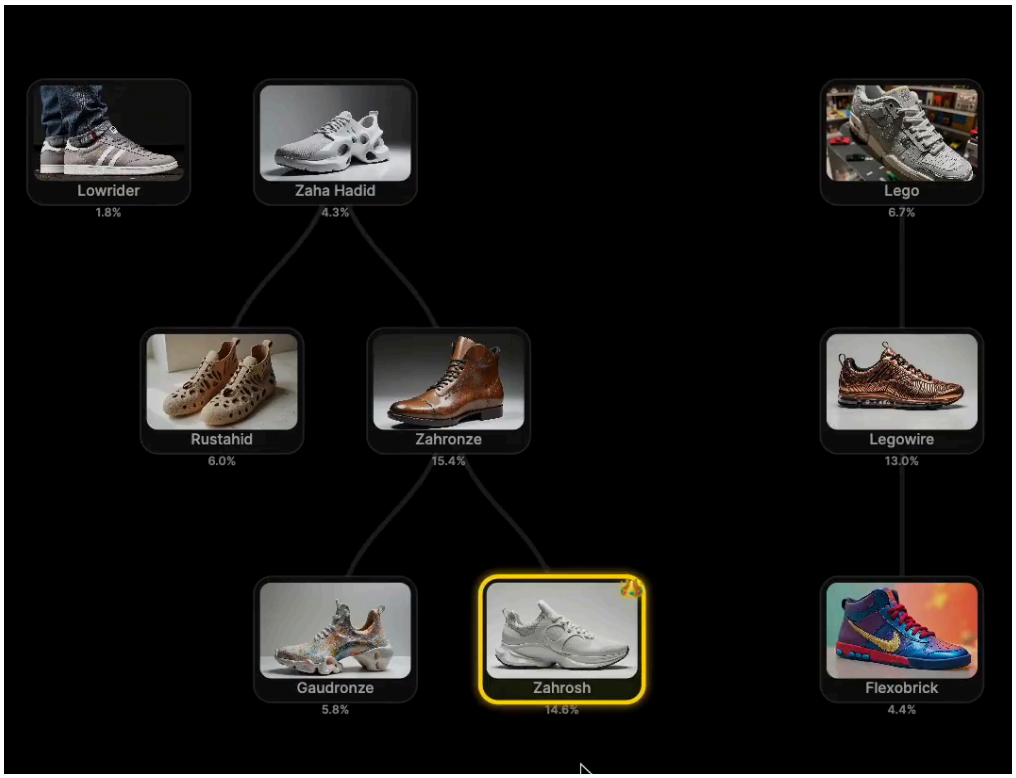

Figure X: Evolution of genes towards the concept of "quiet luxury".

5.3 Evaluation Plan

I will conduct an hour-long within-subjects study to understand how users can use StyleGenome to create new product designs by mixing, mutating, and evolving styles. I will invite ten professional designers with backgrounds in product design, recruited through word-of-mouth and known contacts. Participants will not be exposed to the StyleGenome system prior to the study.

Procedure

Participants will access StyleGenome through a web browser and will be asked to think aloud throughout the study. The procedure is as follows:

- *Introduction (5 minutes).* Participants will be provided informed consent and receive an introduction of StyleGenome's components
- *Reproduction Task (15 minutes).* Participants will be given a detailed design brief and asked to try the different components of StyleGenome (crossover, mutation, and evolution) to reproduce a sneaker design.
- *Free Creation Task (20 minutes).* Participants will freely explore StyleGenome to create their own product designs.
- *Post-Study Interview (15 minutes).* I will conduct a semi-structured interview asking about participants' experiences using StyleGenome and to identify areas for improving the system. Participants will also be asked to complete several standard questionnaires.

Measures

I will measures the following metrics from participants:

- *Human-AI collaboration* measured using questions from [Drawing with reframer] (7-point Likert), evaluating controllability, communication, harmony, partnership, attribution, and ownership.
- *Creativity* measured using the Creativity Support Index [CSI] (7-point Likert), evaluating exploration, inspiration, engagement, expressiveness, tool transparency, and effort/reward tradeoff.

- *Usability* measured using the System Usability Scale [SUS] (7-point Likert).
- *Self-rated final product design quality* (7-point Likert).
- *Self-rated overall experience satisfaction* (7-point Likert).
- *Detailed interaction logs* measuring operations such as adding and removing genes on the canvas, generating images from genes including text and image prompts, performing crossovers of genes, performing mutations of genes, and performing automatic evolution of genes including target descriptions.

5.4 Timeline

My goal is to complete the dissertation by Spring 2026. My proposed schedule is as follows:

- *Nov 2025 - Dec 2025:* Continue building the system, including adding detailed logging and conducting pilots.
- *Nov 2025 - Jan 2026:* Conduct user evaluation study, as outlined in the evaluation plan.
- *Dec 2025 - Feb 2026:* Write the paper and aim for a Jan 2026 submission.
- *Jan 2026 - Apr 2026:* Write the thesis and prepare for thesis defense.

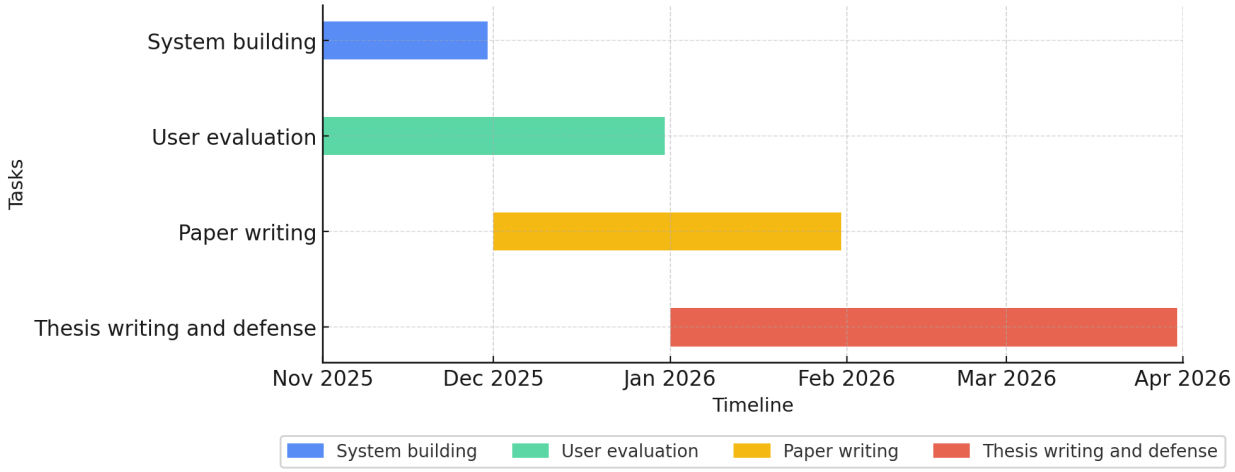

Figure X: Gantt chart of proposed timeline.